Scripts Reference Manual

Copyright © 1999 - 2015 Elipse Software Ltda. All rights reserved.
Version 4.6.161 (01/09/2015)

Table of Contents

1 Introduction 1
1.1 Objects 1
1.2 Scripts 2
1.3 Picks 7
1.4 User-Defined Events 13
2 Programming in E3 16
2.1 Programming Environment 16
2.2 Declaring Variables 17
2.3 Getting References to Objects 17
2.4 Accessing Objects 36
2.5 Working with Collections 37
2.6 Set Command 38
2.7 E3Globals 39
2.8 General Events, Methods, and Properties of Objects 46
3 User Libraries 68
3.1 XControls and XObjects 68
3.2 ElipseX Properties 70
4 View 73
4.1 Viewer 73
4.2 Frames and Splitters 100
4.3 Screens and Screen Objects 107
4.4 E3Alarm 223
4.5 E3Browser 238
4.6 E3Chart 245
4.7 E3Playback 291
4.8 Reports 292
5 Server Objects 331
5.1 Common Properties 331
5.2 Collection of Alarm's User Fields 333
5.3 Server's Runtime Objects 336
5.4 Configuration-Only Objects 339
5.5 Drivers 341
5.6 Data Server 397
5.7 Database 419
5.8 Historic 422
5.9 Storage 426

5.10 Formulas 433
5.11 Alarms 438
6 Frequently Asked Questions 463

CHAPTER 1

Introduction

Scripts are programming language modules that allow users to create procedures associated to specific events, providing them with higher flexibility when developing applications. Every E3 object (item of an application) has a list of previously defined events, but it is also possible to define new user-created events.

1.1 Objects

Objects are reusable software components that allow maximizing usage, and increasing quality and productivity of applications.

An E3 object encapsulates or contains three different parts (properties, methods, and events) that users can manipulate to obtain the advantages of its functionality in their application.

Properties define the object's attributes, such as the appearance of a Screen control, or an object's initial value when users start the application.

Methods are functions that perform a specific action with or within an object.

Events are notifications generated by an object as a response to some particular occurrence, such as a mouse click or a change in a Tag's value, among others.

One of the main features of objects and object-oriented languages is their ability of inheritance among each other, that is, they can inherit features of one or more objects, having the same specific functionality. So, users can have different objects working together to provide features of another derived object.

For example, take the object E3Chart. It consists internally of many objects, such as titles, legends, scales, divisions, queries, and pens. Notice that each object contributes for E3Chart's functionality as a whole: scales help locating point values, legends help identifying pens and their values, and pens help drawing values in the E3Chart.

By handling objects within the E3Chart, users can create two instances of this object that are very different from one another. To handle a specific object, users must access it through a hierarchy. If both E3Charts are on the same Screen, users must first access the Screen, then the preferred E3Chart, and then one of its properties or child objects. When there are many objects of the same type, usually they can be accessed through a collection. A collection is a special object that manages a set of similar objects. An example in the E3Chart is the Pens collection, which allows access to all E3Chart pens.

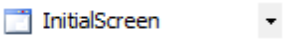
1.2 Scripts

The language that E3 Studio uses in its scripts is **VBScript**, a subset of the Visual Basic® language developed by Microsoft. VBScript has a fast, light, and portable interpreter, developed for use on Internet browsers and other applications using ActiveX Controls, Automation Servers, and Java Applets.

As seen before, scripts are linked to events of an object. However, to facilitate and increase development speed, E3 has already incorporated some of the most common actions that can be performed with scripts, through wizards known as **Picks**. Users can therefore define that a given event will either run a script, a Pick, or a combination of both, in a sequence that is also pre-defined.

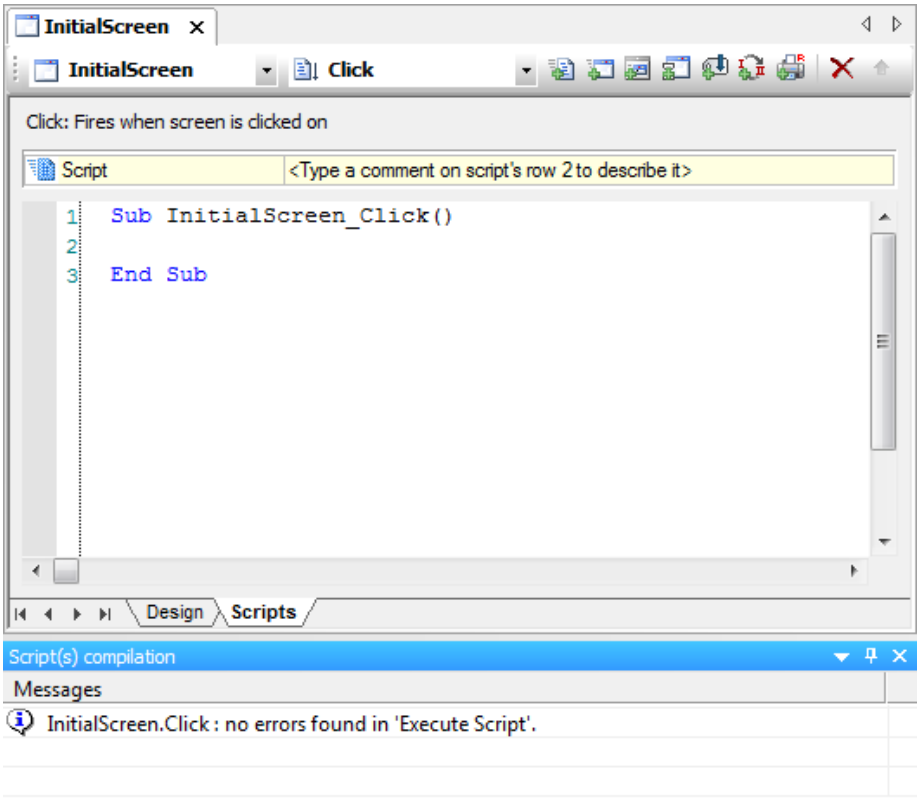
Each view in E3 Studio displays at least two tabs on its lower side: **Design** and **Scripts**, except for Database and Alarm Server objects, which do not have a **Design** tab. Objects and their child objects can be manipulated on the **Design** tab; to manipulate their scripts, use the **Scripts** tab. The available options on the latter are:

Available options for Scripts tab

ICON	OPTION	DESCRIPTION
	**List of Objects**	Selects the object whose script will be manipulated.
	List of Events	Selects the event to apply to an object.
	Script	Adds scripts linked to the event.
	Pick Open Screen	Adds an **Open Screen** Pick.
	Pick Open Modal Screen	Adds an **Open Modal Screen** Pick.
	Pick Run Application	Adds a **Run Application** Pick.
	Pick Load Value	Adds a **Load Value** Pick.
	Pick Toggle Value	Adds a **Toggle Value** Pick.
	Pick Print Report	Adds a **Print Report** Pick.
	Remove selected script or Pick	Removes the selected script or Pick from the Actions List.

ICON	OPTION	DESCRIPTION
	Move selected script or Pick up	Moves the selected action up, in the sort order of the Actions List for the event.
	Move selected script or Pick down	Moves the selected action down, in the sort order of the Actions List for the event.
	AppBrowser	Opens the AppBrowser window.
	Find	Finds occurrences of a given text.
	Find previous	Selects the previous text occurrence in the results list.
	Find next	Selects the next text occurrence in the results list.
	Replace	Replaces the occurrences in the results list by another specified text.
	Create user event	Creates a user-customized event.
	Remove user event	Removes the selected user-customized event.
	Edit user event	Edits the selected user-customized event.
	Compile selected script	Compiles the selected script, and shows its errors on the **Messages** panel.
	Compile all scripts from this event	Compiles all scripts linked to the event.
	Compile all events from this object	Compiles all events linked to the object.

The execution order of actions is top-down. To change this order, use the and options. Use the option to check for errors in the specified script for the event. The compiler's error messages are displayed on the **Messages** panel, which can appear as a floating window or docked on the lower or upper side of the **Scripts** tab. Double-click the error to select it in the script.

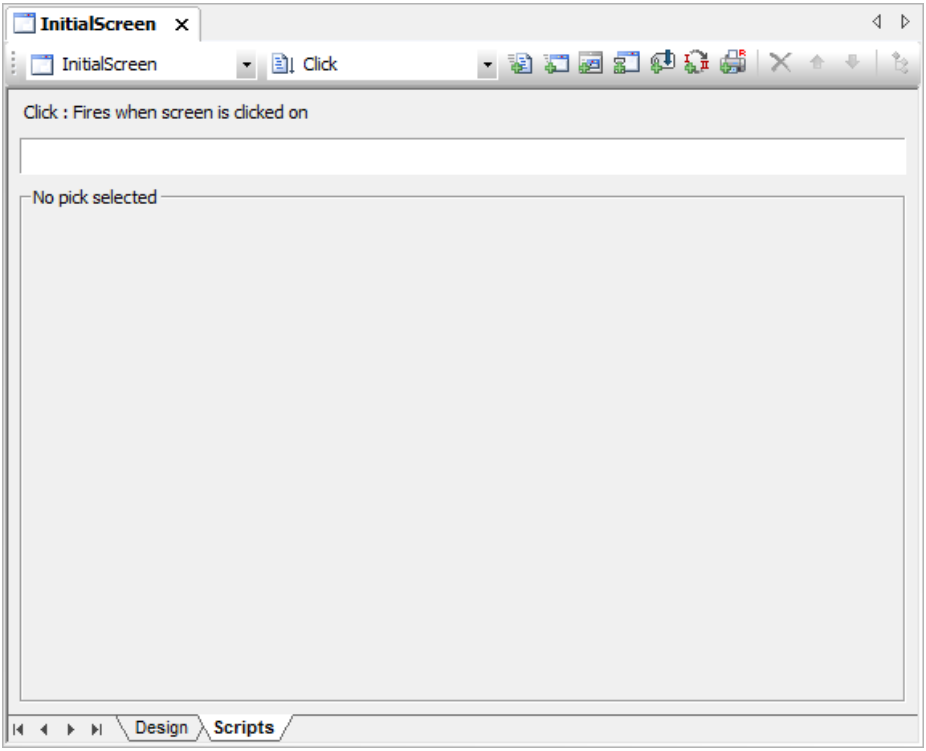


Compiler messages

1.2.1 Adding a Script

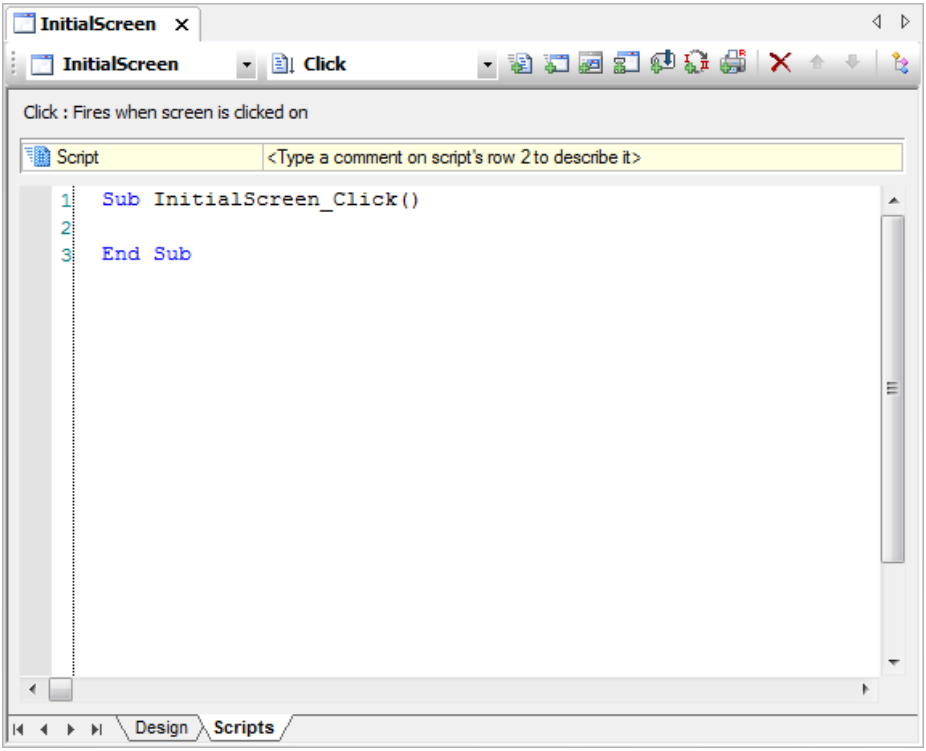
To add a script to an object, follow these steps:

1. Select the object to create the script, and then click the **Scripts** tab.



Scripts tab

2. Click the icon. The Scripts Editor is then opened, according to the next figure.



Adding a script to an object

3. Type the VBScript commands in the text-editing box.

NOTE: Use the underline character to add a line break, to make code more legible. The underline character indicates that the code continues on the next line.

For example:

```
If intBoilerTemperature3 > 120 and _
  intBoilerTemperature4 > 120 Then
  bSendAlarm = True
  bAlarmOn = True
End If
```

Each event can have several scripts and Picks linked to it, called **Event Actions**. The list of actions can be seen on the upper side of the scripts editing window. Each object can have any number of events with linked scripts or Picks.

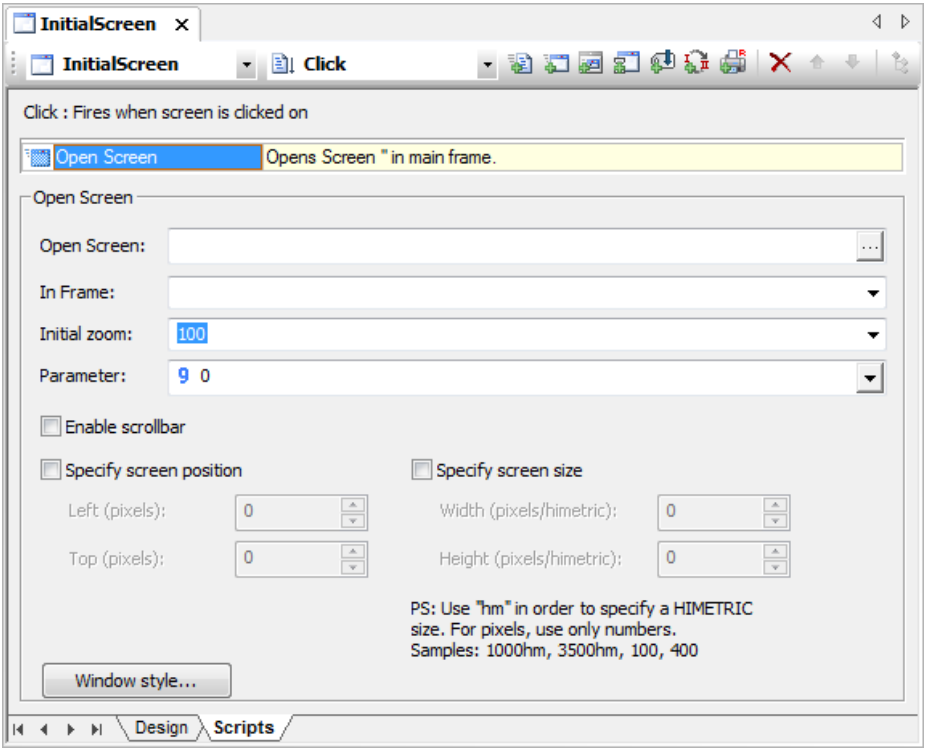
NOTE: When right-clicking any of these actions described previously, a contextual menu with options to cut, copy, and paste scripts and Picks among events is then opened.

1.3 Picks

Picks implement a friendlier way of performing the most common procedures, by saving configuration time. Among them, there are actions, such as switching Screens or attributing values, which are very common while creating a project. Next, there is a description of the available Picks on the **Scripts** tab.

1.3.1 Open Screen

Opens a specific Screen or Frame.



Settings for the Open Screen Pick

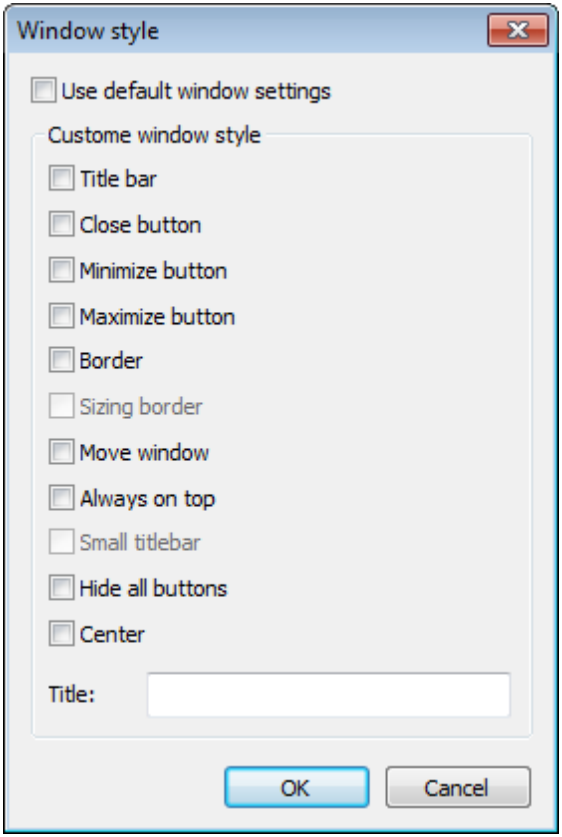
Available options for Open Screen Pick

OPTION	DESCRIPTION
Open Screen	Selects the Screen to open.
In Frame	Selects the Frame where the Screen will be viewed. If it is blank, it will be the main Frame (top).
Initial zoom	Defines the Screen's zoom, when displayed.

OPTION	DESCRIPTION
Parameter	Indicates a parameter to pass when calling the Screen.
Enable scrollbar	Allows using scroll bars on the Screen.
Specify screen position	Indicates the Screen position, in pixels.
Specify screen size	Indicates the Screen size, in pixels or Himetric.
Window style	Opens the **Window Style** dialog box.

1.3.1.1 Window Style Dialog Box

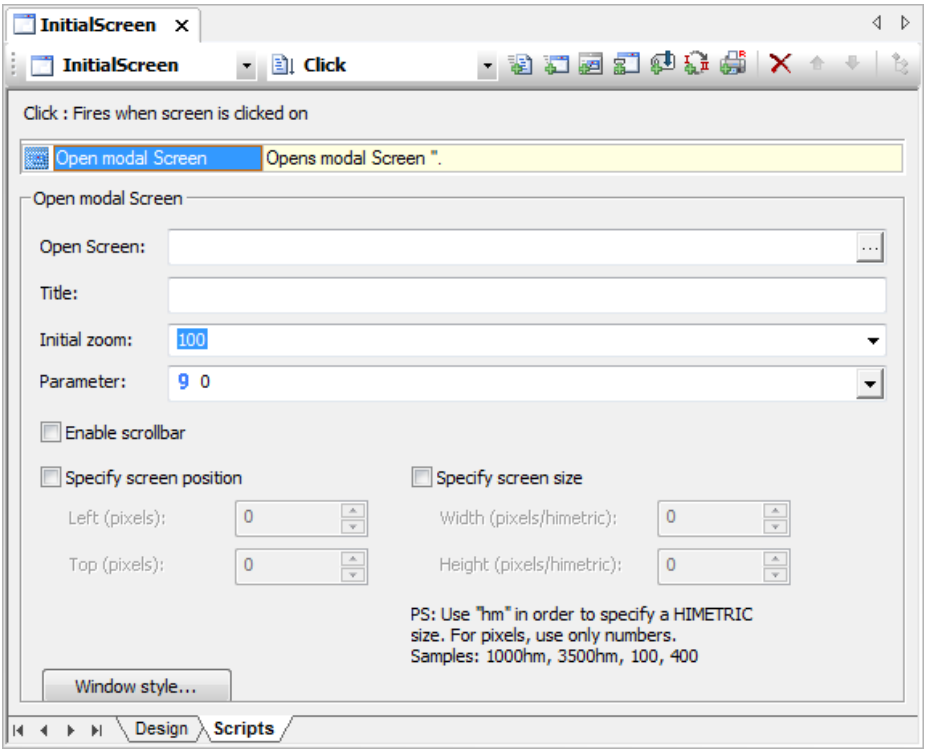
Allows users to set the style of the window to display, by defining title and availability of borders, and the close, maximize and minimize buttons, among other settings. If the **Use default window settings** option is selected, the system disables these window options and assumes the Viewer's default style, as seen on **Viewer** tab, under Viewer properties.



Window Style dialog box

1.3.2 Open Modal Screen

Opens a Modal Screen, that is, a Screen that does not allow user interaction with other Screens while it is active.



Settings for the Open Modal Screen Pick

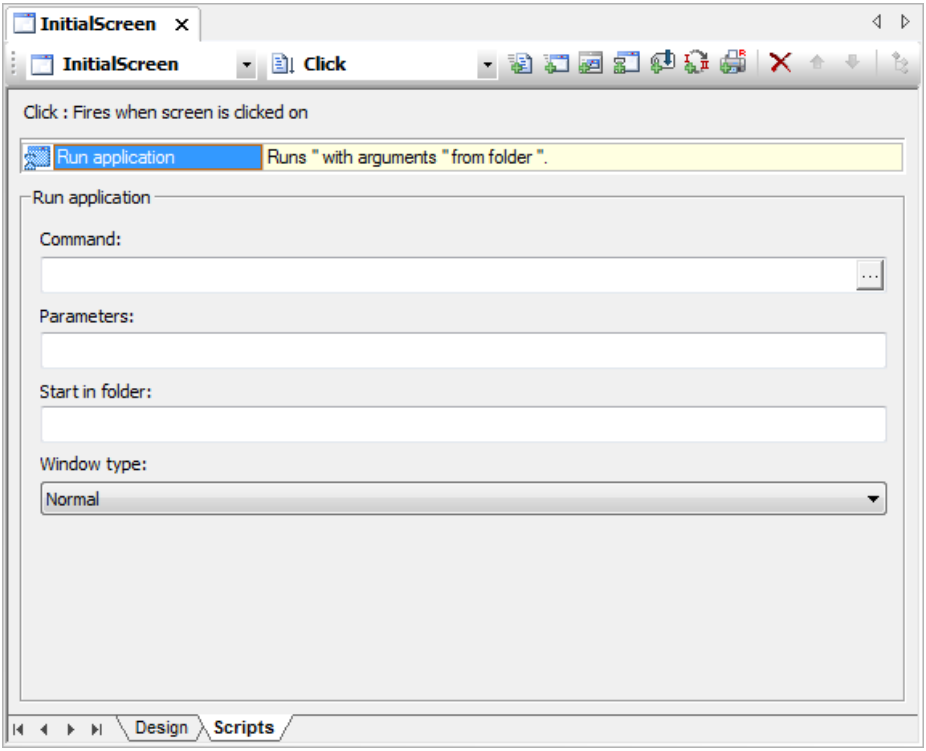
Available options for Open Modal Screen Pick

OPTION	DESCRIPTION
Open Screen	Selects the Screen to open.
Title	Defines the window title. This text is concatenated to the Screen's name.
Initial zoom	Defines Screen's zoom, when displayed.
Parameter	Indicates a parameter to pass when calling the Screen.
Enable scrollbar	Allows using scroll bars on the Screen.
Specify screen position	Determines the frame's position, in pixels, on the Screen, starting at Screen's upper left side.
Specify screen size	Determines Screen's width and height, in pixels or Himetric.

OPTION	DESCRIPTION
Window Style	Allows configuring the style of the window to display, by defining title and the availability of borders and the close, maximize and minimize buttons, among other settings (check the **Window Style Dialog Box** topic).

1.3.3 Run Application

Executes a specific application.



Settings for the Run Application Pick

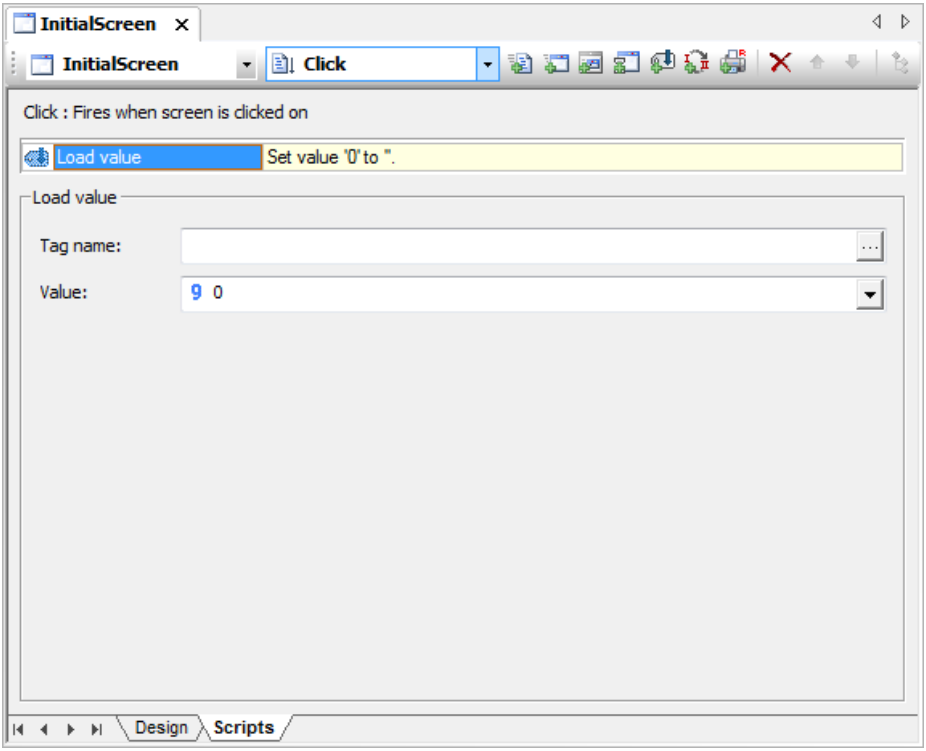
Available options for Run Application Pick

OPTION	DESCRIPTION
Command	By clicking ..., it is possible to browse the disk to select an application file to run.
Parameters	Allows specifying arguments for application call.
Start in folder	Determines the working directory of the application to run.

OPTION	DESCRIPTION
Window type	Determines the type of the execution window for the application: **Normal**, **Minimized**, or **Maximized**.

1.3.4 Load Value

Loads a value to a Tag.



Settings for the Load Value Pick

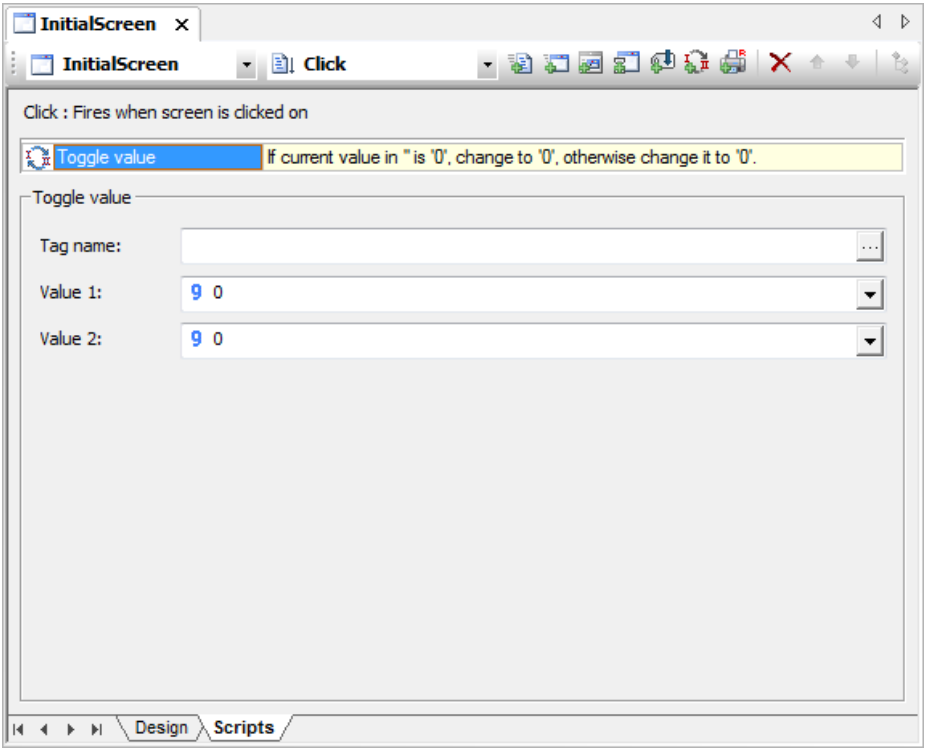
Available options for Load Value Pick

OPTION	DESCRIPTION
Tag name	Specifies the Tag name to which the value will be loaded. It is possible to select the Tag in the AppBrowser, by clicking ...
Value	Determines the value to load to the Tag. It is possible to select a data type, by clicking the arrow key.

1.3.5 Toggle Value

Allows toggling a Tag value. If the Tag value is equal to *Value1*, then the Tag receives *Value2*. If the Tag value is equal to *Value2*, then the Tag receives *Value1*. If the Tag value is different from either *Value1* or *Value2*, then the Tag receives *Value1*.

It is possible to add as many **Toggle Value** Picks as needed. This allows checking multiple values for the same Tag, or even for several Tags in the same event.



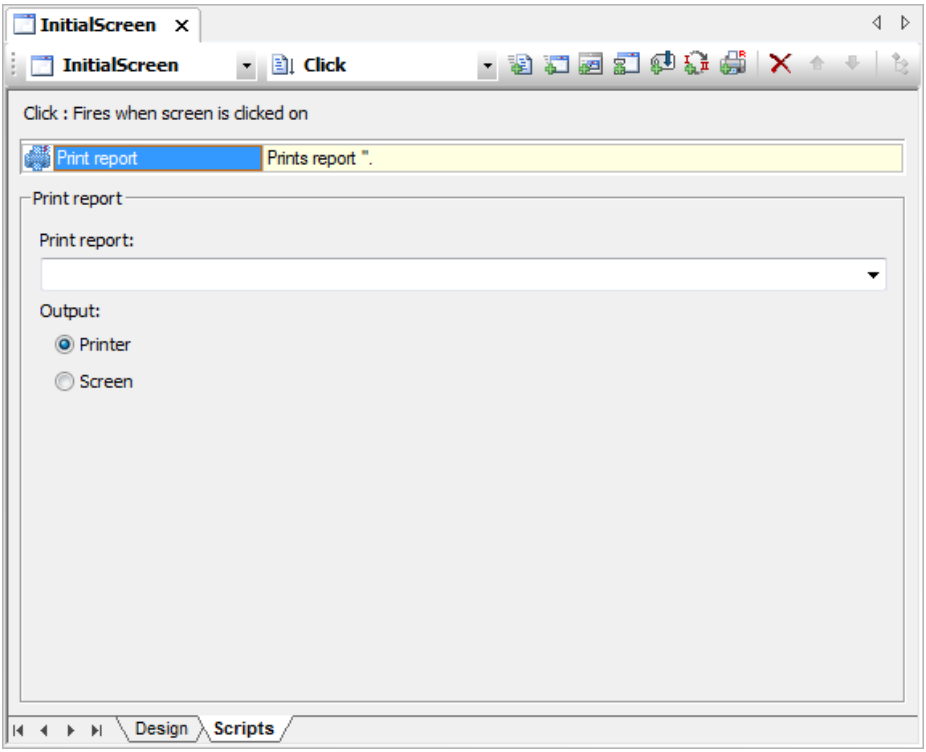
Settings for the Toggle Value Pick

Available options for Toggle Value Pick

OPTION	DESCRIPTION
Tag name	By clicking ..., the AppBrowser is opened to select the preferred Tag.
Value 1	Determines the first value to compare. If the Tag value is equal to *Value1*, then the Tag receives *Value2*.
Value 2	Determines the second value to compare. If the Tag value is equal to *Value2*, then the Tag receives *Value1*.

1.3.6 Print Report

Allows printing a Report on a Screen or to a printer.



Settings for the Print Report Pick

Available options for Print Report Pick

OPTION	DESCRIPTION
Print report	Allows choosing the report to print.
Output	Determines the output type of the report: • **Printer**: Sends the report to a printer. Corresponds to the **Print** method • **Screen**: Performs a printing preview of the report on screen. Corresponds to the **PrintPreview** method

1.4 User-Defined Events

Although E3 comes with a wide range of events, there are times when users may wish to create a specific event for their applications.

An example of using user-customized events would be a calculation, or a more

complex task in an object, when the generated event comes from another Tag or property.

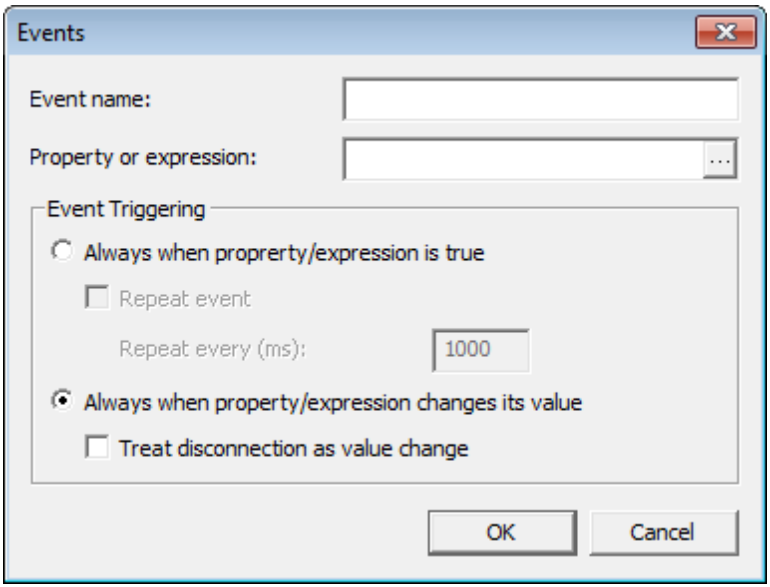
Although users can create and execute this type of work from the Tag or property generating the event, there are several advantages in keeping the script next to the object that will suffer script's action. Among these, there is the additional work needed to make an object point to another one, not to mention the small amount of work needed in maintenance, because if by any reason it is necessary to modify or delete a Tag or property generating the event, it will not be necessary to modify a second object.

Another advantage comes from the fact that if users erase the Tag generating the event, the object does not lose its script, as long as users indicate another generating source for the event.

The generation of internal events also makes the creation of libraries easier, because each time a library component is inserted in an application, it also brings scripts and calculations that can be necessary, reducing the configuration work.

To generate a new internal event in an object, follow these procedures:

1. Click **Create user event** , or else **Create new event** , on the Events List. E3 then opens a window to define event properties.



Window for adding user-customized events

Available options for Events window

OPTION	DESCRIPTION
Event name	Name that identifies the event.

OPTION	DESCRIPTION
Property or expression	Expression generating the event. It can be copied using the AppBrowser, by clicking ...
Always when property/expression is true	Indicates whether the event is an **etOnEvent**-type (the event occurs whenever the expression is true) or an **etWhileEvent**-type (the event occurs cyclically, at pre-defined intervals).
Repeat event	When filled in, indicates that it is an **etWhileEvent**-type event. Indicates the event's repetition cycle, that is, its periodicity while the generating expression is true, in milliseconds.
Always when property/expression changes its value	Indicates that it is an **etOnValueChanged**-type event, that is, the event occurs whenever the generator expression changes its value.
Treat disconnection as value change	Indicates whether a connection or disconnection of the expression generating the event must be considered as a change.

2. Click **OK** to finish the process and insert the event. It will now appear on the events list.
3. To change this event, select it and click **Edit user event** . The previous window is then opened again to edit event data.
4. To delete this event, select it and click **Remove user event** .

IMPORTANT: When clicking **Remove user event**, scripts for this event are lost.

CHAPTER 2

Programming in E3

Although the majority of VBScript aspects apply to scripts programming with E3, some particularities must be emphasized regarding the implementation of the concept of object orientation on the system.

VBScript is a language based on Visual Basic that brings the capacity of scripting to applications that run on Windows.

VBScript exchanges information with applications using the *ActiveX Scripting* technology. With ActiveX Scripting, browsers and other client applications, such as Viewers, can compile scripts and call functions, among other procedures. This enables scripts developed for an application or library, which can be executed in a graphical interface, to run either in Viewer or in an Internet browser, with no need to adapt it on the application.

Further information about VBScript can be obtained on the **VBScript Reference Guide**, in the Program Group **Elipse E3**.

2.1 Programming Environment

The programming environment for scripts in Studio can be accessed by right-clicking any object, and then selecting the **Properties** option. On **Scripts** tab of the object view, users can see a combo box where they can define which event will be the script's generator. As seen in the previous chapter, there are two types of events in an E3 object: pre-defined events, and user-customized events.

Pre-defined events vary from object to object, depending on its usage and functionality. A Screen object, for example, has graphical-interface-related events, such as **Click** (called when clicking the object) or **DbClick** (called when double clicking it); an object such as an I/O Driver, on the other hand, has communication-related events, such as **OnCommError** (called when there is a communication error). Users can also define other events for the object, as seen before.

When users link a script to an event in an object, the text field presents a procedure declaration whose definition is automatic, and formed by the following text:

```
Sub ObjectName_EventName()
End Sub
```

Where `ObjectName` is the name of the linked object, and `EventName` is the name of the aforementioned event. Script commands should be placed between these two lines.

To help typing the script, users can use the AppBrowser. After choosing the preferred method or property, they can click **Copy**. The chosen Tag, property, or method will be

inserted at the cursor position on script's edit view. The cursor position is shown by a blinking insertion point.

2.2 Declaring Variables

Users can declare variables in two different ways: **implicitly** or **explicitly**.

To declare a variable implicitly, simply use its name on the script. The variable will automatically be created and initialized with the attribution value, or it will remain with the EMPTY value, in case it does not receive any value before using it.

This is a quick practice, but if the script is very large, it may cause confusion and creation of more than one variable with the same name, generating bugs in the script.

To declare variables explicitly, use the **Dim** command, as in the next example:

```
Dim Temperature
```

Users can declare multiple variables by separating each variable's name by a comma. For example:

```
Dim Left, Right, Top, Bottom
```

Because all scripts in E3 are linked to a specific object, variables are always local, valid only on script's scope. For public or global variables, users must create an Internal Tag and then use it to store the preferred value.

2.3 Getting References to Objects

One of the most important features to consider when working with scripts inside E3 is the separation between processes executed in the server and those executed in the client's interface (Viewer). To work with scripts, users can manipulate:

- Server objects via server
- Server objects via Viewer(s)
- Viewer objects via the same Viewer

However, users cannot directly manipulate:

- Viewer objects via server (this is only possible by creating events in Viewer, which are connected to variables in the server)
- Viewer objects via another Viewer (this is only possible by creating events connected to variables in the server)

Such limitations come from the fact that, by definition, there is an independence between what the Viewer's station is performing or viewing and the server, and vice

versa. So, all activities, both in the server and in Viewer, need to be coordinated either in an asynchronous way or through events, to operate harmoniously.

Thus, due to this independence, when creating a script, first users should obtain a correct reference to the objects they want to manipulate, that is, it is necessary that the object be found on several E3 modules first.

Remember that when users edit a script, they can use the AppBrowser, which allows the path of a method or property to be completely copied to the script, thus helping to create scripts.

Therefore, to access external objects being manipulated in a script, some basic guidelines are used. For example, if users want to manipulate the value of an I/O Tag, the path is **Server - Driver - Folder (if available) - Tag**. But if their goal is to manipulate a button on a Screen, the path is **Viewer - Frame (if available) - Screen - Button**.

There are basically three source locations for scripts, according to the methodology for accessing objects:

- Server
- Screens and Frames (Viewer)
- ElipseX (libraries): they may be XObjects (running on the server) or XControls (running on a Viewer)

2.3.1 Accessing Server Properties

To access an object running on the server from a Screen object or an ElipseX, use the **Application.GetObject** method.

The word **Application** returns the application object related to the current context of the object, and the **GetObject** method searches for an object with the given name within **Application**, on the server. Example:

```
Sub Button1_Click()
  Application.GetObject("Driver1")._
    Item("tag001").AllowRead = False
End Sub
```

Or else:

```
Sub Button1_Click()
  Application.GetObject("Driver1.tag001").AllowRead = False
End Sub
```

The **Item** method was used to locate `tag001` from a reference to `Driver1`, because the Driver is a collection of Tags. After locating the object, its properties and functions can be freely accessed.

In case users need to accomplish another operation with `Driver1` or `tag001`,

another alternative for the previous script is:

```
Sub Rectangle1_Click()
  Set obj = Application.GetObject("Driver1")
  obj.Item("tag001").AllowRead = False
  obj.Item("tag002").AllowRead = False
End Sub
```

In this case, the variable `obj` is pointing to the object `Driver1`, and the next time users want to access some object descending from `Driver1` inside the script, they can use the variable `obj` directly. This brings a performance enhancement, since every call to the **GetObject** method accesses the server only once. With this technique, users avoid unnecessary calls to a server. This example uses the **Set** command, explained later. Notice that the usage of variables also makes code clearer and more easily modifiable. In case it is necessary to change the object where users want to execute commands, just change the attribution line of this variable.

The word **Application** in scripts can indicate either functions executed on a Viewer or on the Server. In this case, the Application object knows beforehand which functions should be executed in each case. However, users cannot execute Viewer functions inside the server, and it is not possible to execute server functions inside a Viewer.

2.3.2 Accessing Studio Properties

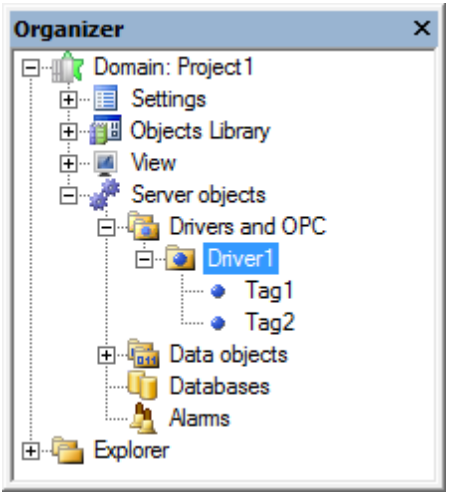
Accessing any server object in a script that runs in Studio can be performed by using the **Application.GetObject** directive. The word **Application** returns the application object related to the current object's context, and the **GetObject** method searches on the load Domain in Studio for a server object with the provided path. Example (the **CustomConfig** event is triggered in Studio):

```
Sub XControl1_CustomConfig
  Application.GetObject("Data.DemoTag1").DocString =
"Documentation"
End Sub
```

2.3.3 Accessing Server Properties from the Server

Take the example of accessing Tag properties from another Tag, origin and destination are on the server, even though connected via a `Driver1`'s parent module.

In this case, the **Parent** declaration must be used. This allows first accessing the parent object where the script is, and then users traverse the hierarchy to look for another element.

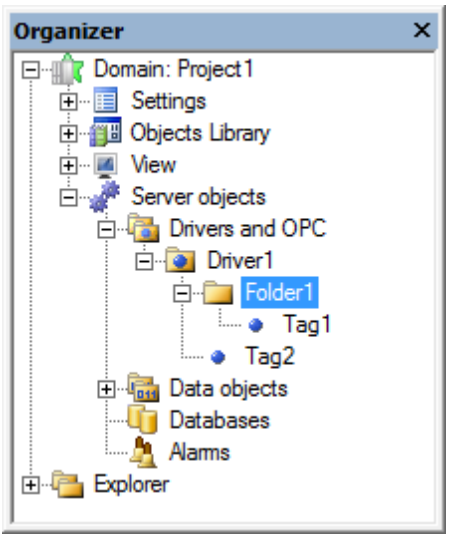


Driver1 is the parent object of Tag1 and Tag2

Example:

```
Sub Tag1_OnRead()
  Parent.Item("Tag2").AllowRead = False
End Sub
```

If users are inside a group and they want to access the same Tag2, they can nest various **Parent** commands.



Folder1 is the parent object of Tag1

Example:

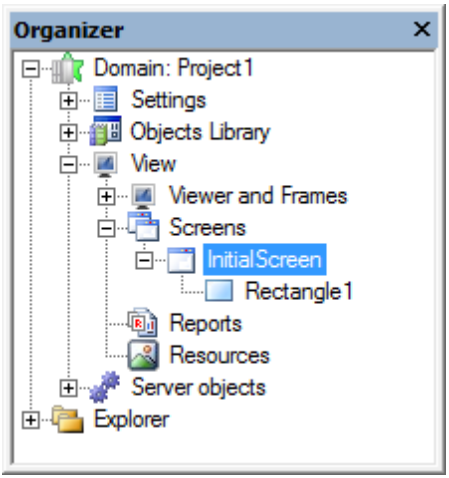
```
Sub Tag1_OnRead()
```

```
  Parent.Parent.Item("Tag2").AllowRead = False
End Sub
```

2.3.4 Accessing Screen Objects from a Screen Script

In this case, users must only use the **Item** method, as the objects are children of the Screen. For example:

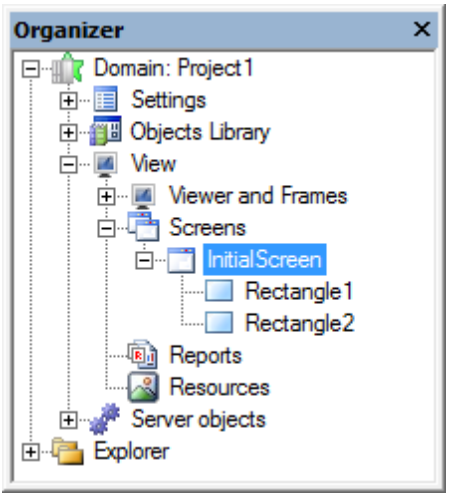
```
Sub Screen1_OnPreShow(vArg)
  Item("Rectangle1").Visible = True
End Sub
```



Rectangle1 is an item of InitialScreen

2.3.5 Accessing Screen Objects from a Script on Another Object Screen

In this case, users can use the **Parent** or the **Screen** method.



InitialScreen is the parent object of Rectangle1 and Rectangle2

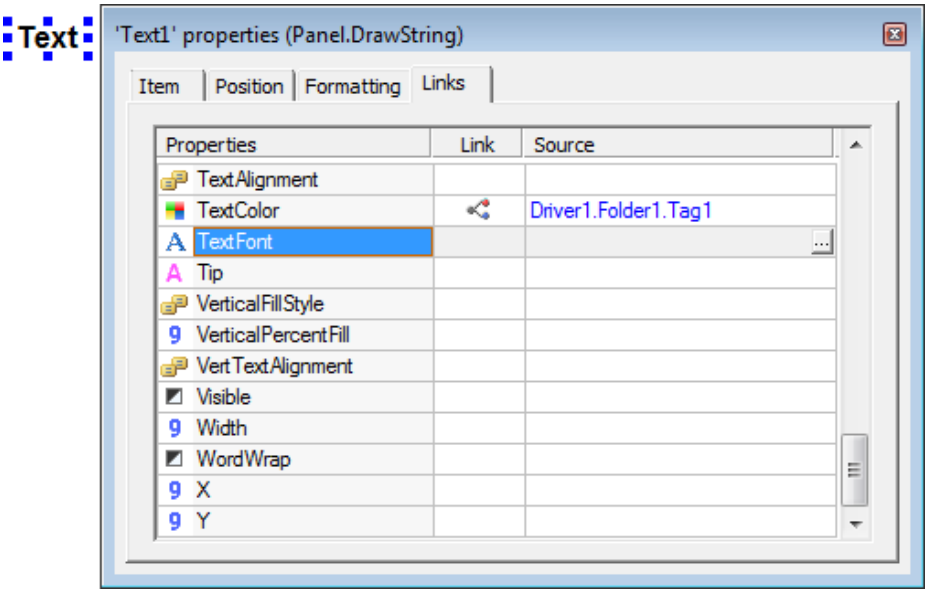
Example:

```
Sub Rectangle1_Click()
  Parent.Item("Rectangle2").Visible = True
End Sub
```

2.3.6 Changing a Screen or Screen Objects from the Server

The modification of any behavior on a Screen can be performed using Links (the server automatically reports all changes in the chosen variables to Viewers), or by an explicit Viewer's search for information in the server. The whole graphical interface linking operation is accomplished from client to server and not from server to client. Thus, it is not possible to change Screens or objects from the server by using scripts, because each data client is a different copy of the Screens.

A practical example is either to change the color of a text in the Screen to green when a Tag is turned on (value one) or to red when it is turned off (value zero). In this case, simply create a Digital Link between `Text1`'s **TextColor** property and `Tag1`. Links are preferable due to execution speed and also simplicity in maintaining and building applications.

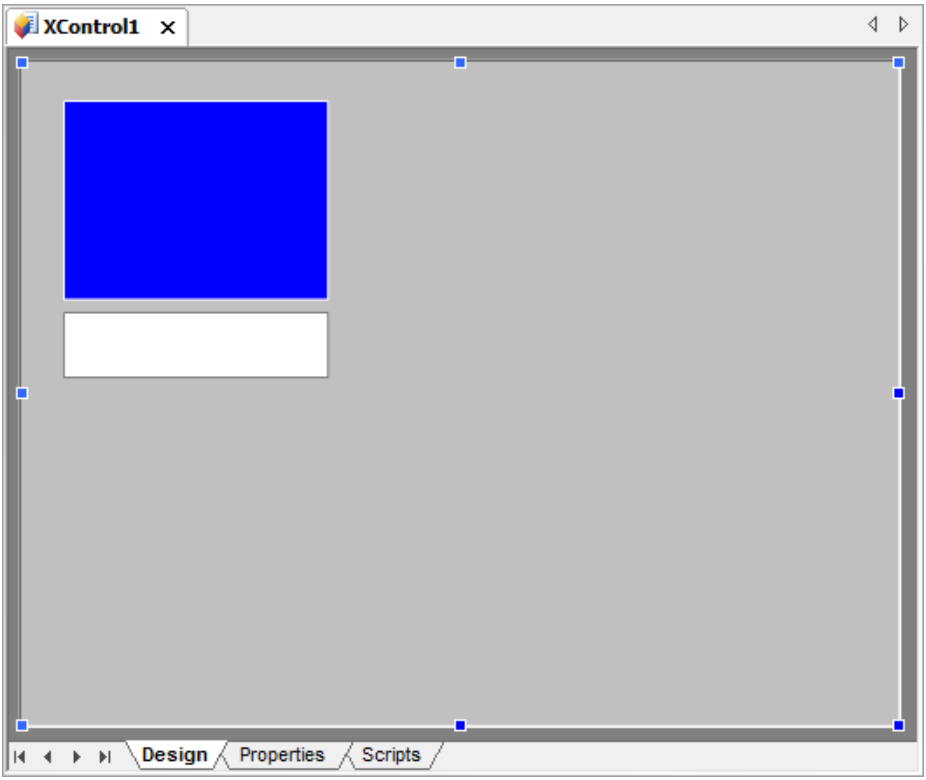


Linking text color to Tag1's value

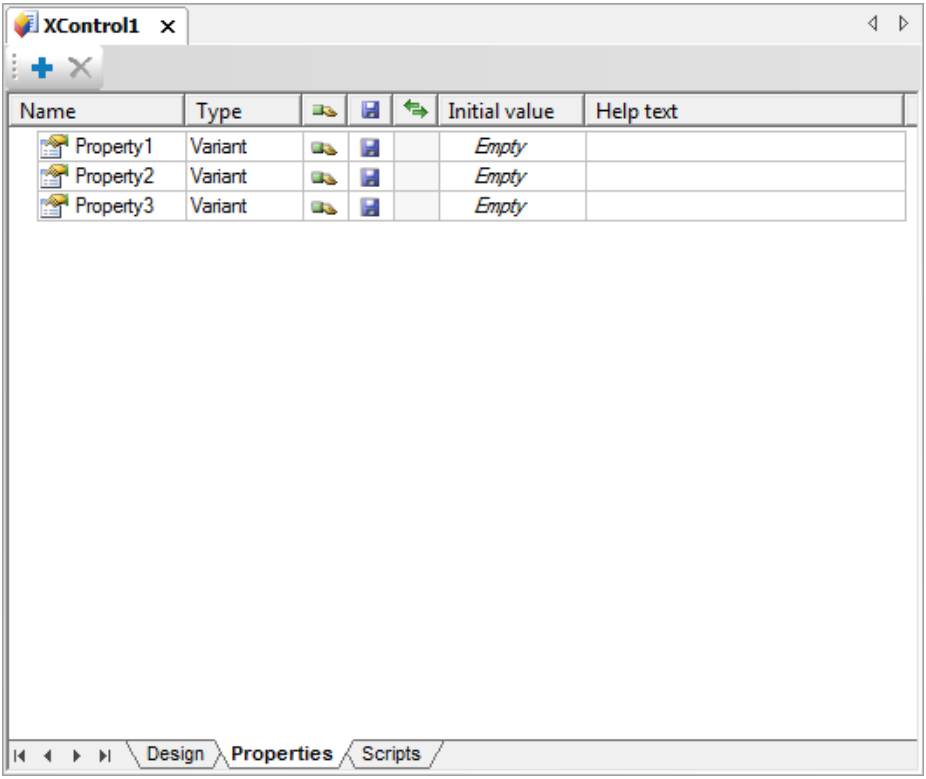
Another way to execute the previous procedure is by creating a script in the Viewer that constantly checks whether Tag1 changed its value, and then changes the text color. This type of script can be accomplished, but it strongly degrades application performance. Thus, this practice is **not** recommended.

2.3.7 Accessing ElipseX Objects from the ElipseX Itself

When creating an ElipseX, users can declare properties (XProperties) and insert objects, which can be Screen objects (XControls) or server objects (XObjects). To access XProperties via scripts, just use the property's name directly.



Design tab



Properties tab

For example, in the previous figure there is an XControl1 with **Property1** property, and the objects Text1 and Rectangle1.

The **Property1** property, which is a **Boolean**-type, can be accessed as in the following script:

```
Sub XControl1_OnStartRunning()
  XControl1.Property1 = True
End Sub
```

Or else:

```
Sub XControl1_OnStartRunning()
  Property1 = True
End Sub
```

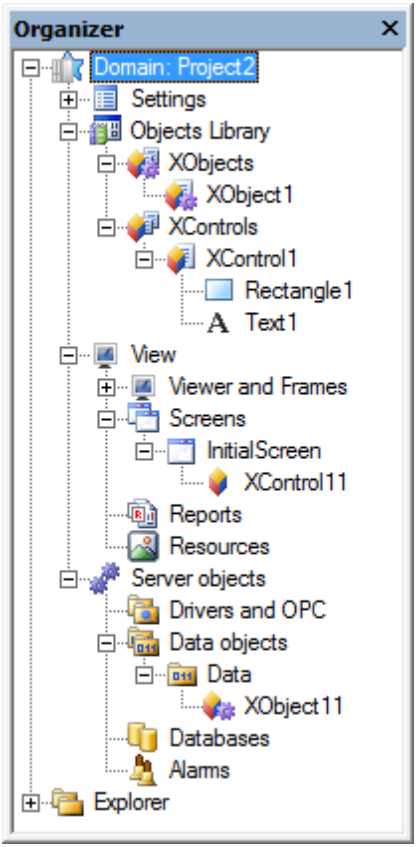
If the ElipseX has any internal objects, then it is possible to use the **Item** method to get references for these objects. For example:

```
Sub XControl1_OnStartRunning()
  Item("Text1").Value = "engine"
  Item("Rectangle1").ForegroundColor = RGB(212, 208, 20)
End Sub
```

2.3.8 Externally Accessing Objects of an ElipseX

An ElipseX object can only be accessed externally via its properties, by using its created instances. It is not possible to access internal objects directly.

If an ElipseX is an XControl, it behaves as a Screen object. For example, consider the application on the next figure.



XControl (example)

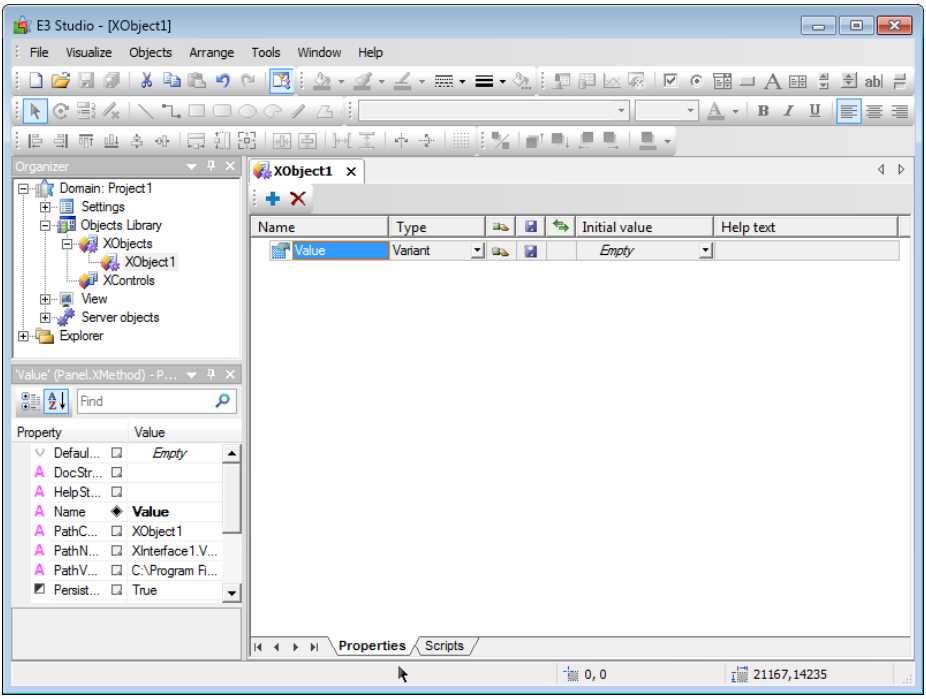
To change XControl's **Property1** property, users can write the following script on a Button's **Click** event:

```
Sub CommandButton1_Click()
  Screen.Item("XControl11").Property1 = True
End Sub
```

Or else:

```
Sub CommandButton1_Click()
  Parent.Item("XControl11").Property1 = True
End Sub
```

In case of an XObject, insert it in a Data Server.



XObject (example)

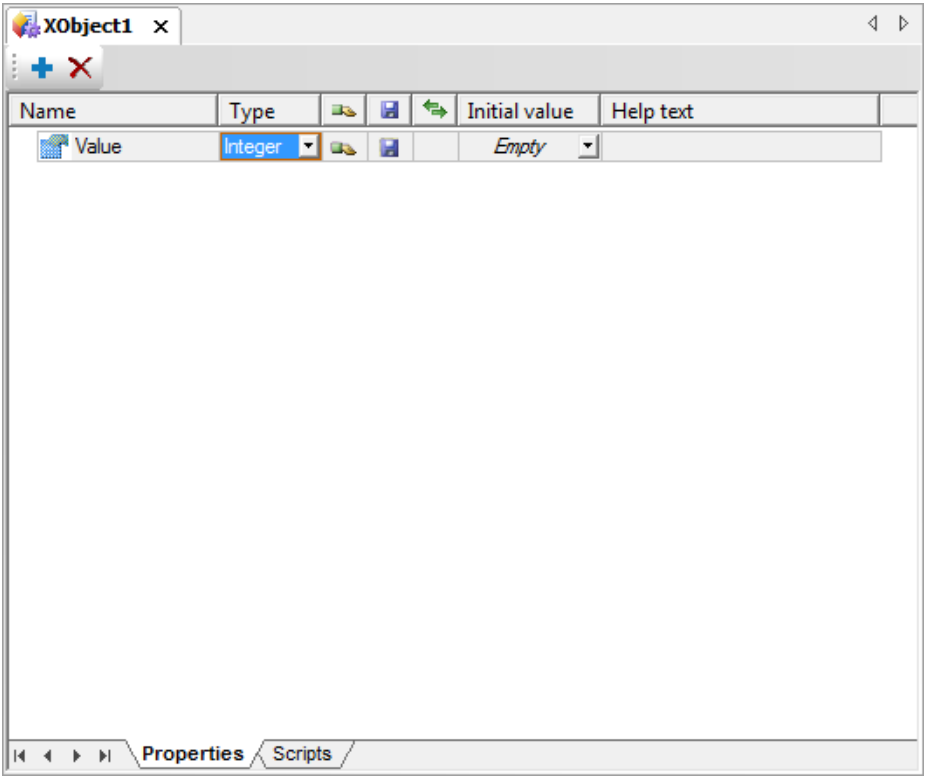
A possible script to alter XObject's **Value** property would be:

```
Sub CommandButton1_Click()
  Application.GetObject("Data.XObject11").Value = 123
End Sub
```

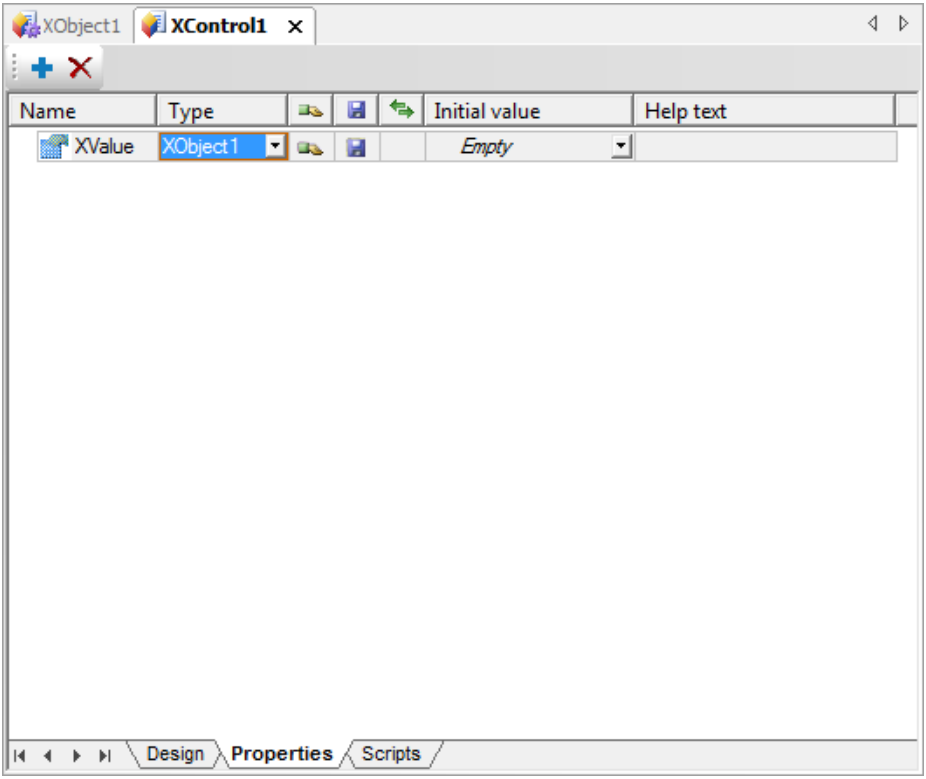
Or else:

```
Sub CommandButton1_Click()
  Application.GetObject("Data").Item("XObject11").Value = 123
End Sub
```

Users can also have an XControl accessing an XObject through an XProperty. For example, the next figure shows an XControl called "XControl1" with an **XValue** property. This property's type is **XObject1**, which is also the name of the XObject created.

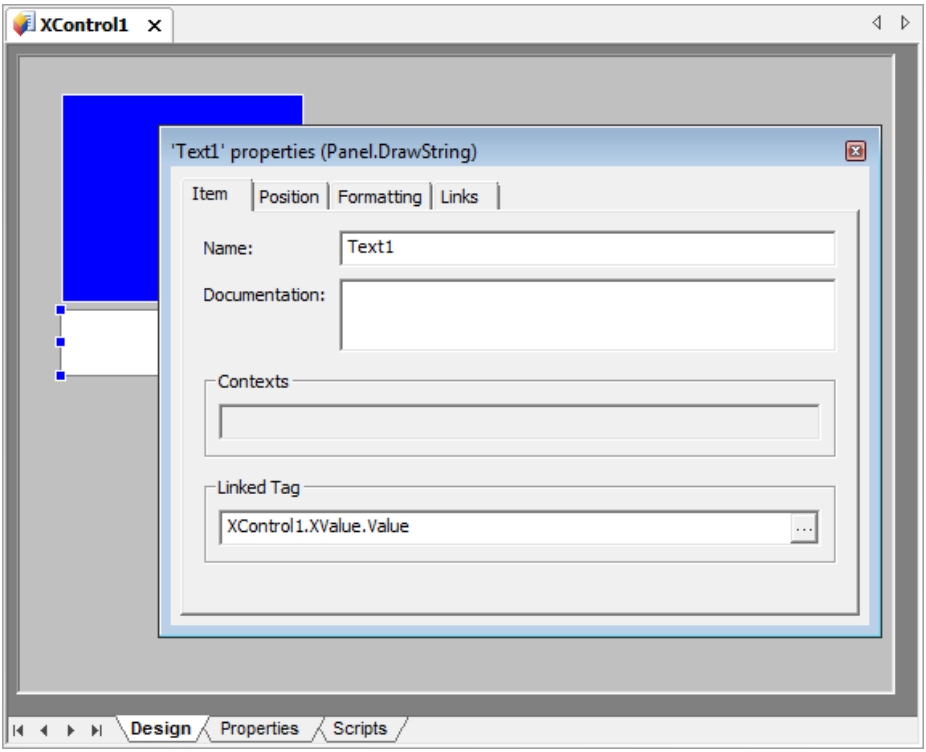

XObject1
Name
Type
Initial value
Help text
Value
Integer
Empty
Properties
Scripts

XObject



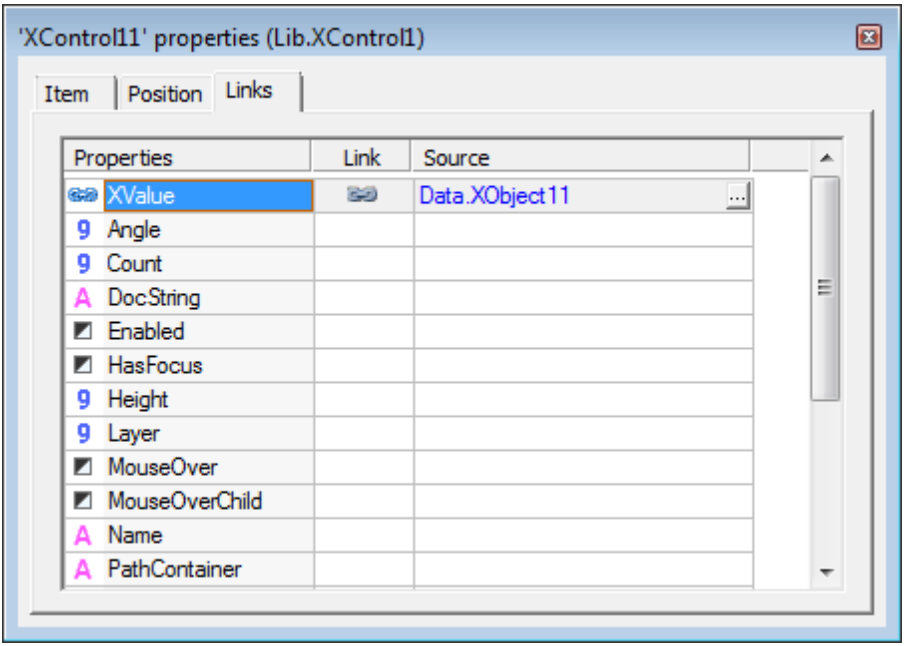
XControl

For example, users can link the value of object "Text1" with XObject1's **Value** property. This is performed via **XValue** property, created in "XControl1". Thus, the value of XObject1's **Value** property is displayed in object "Text1" of "XControl1".



Value property

In this project, the link of an "XObject11" instance can be linked to an "XControl11" instance using a Link in the **XValue** property.

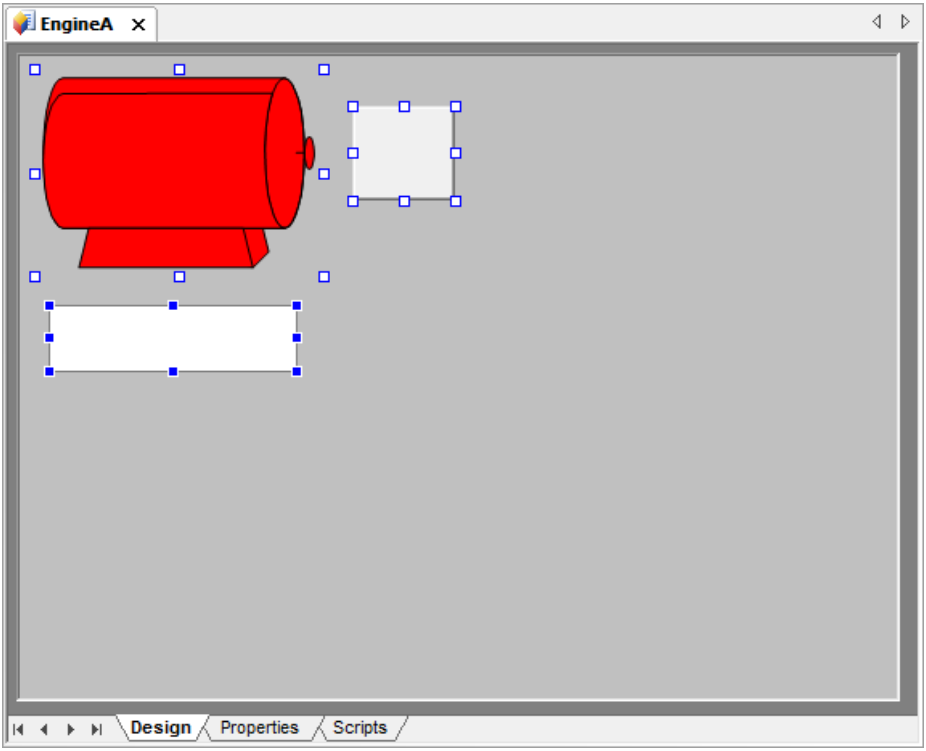


XValue (Link)

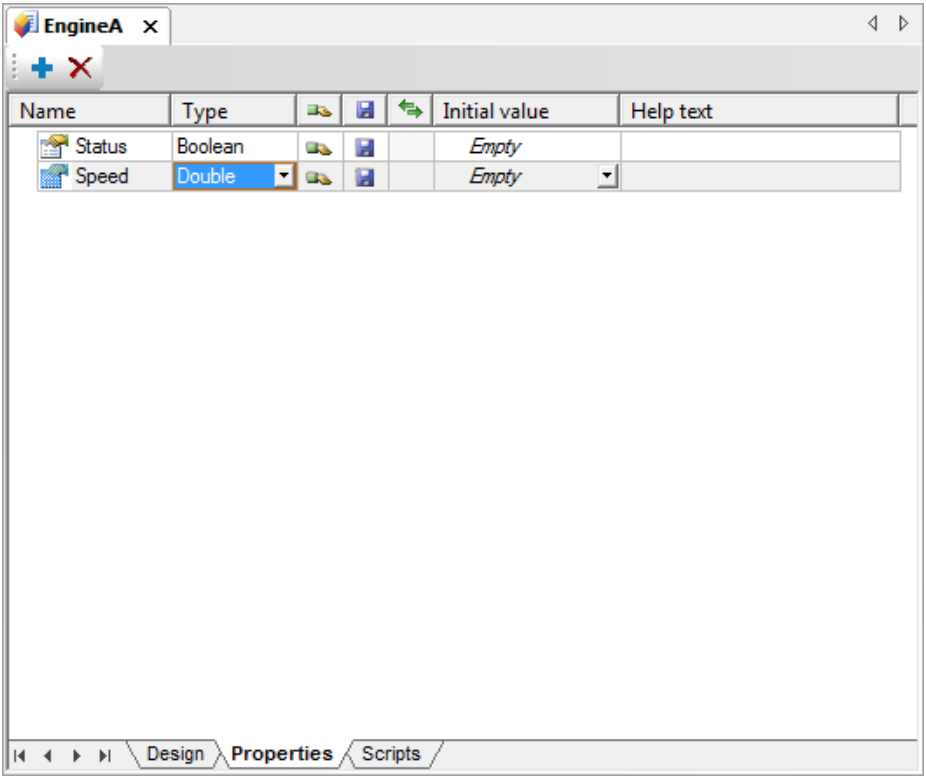
2.3.8.1 Example of Creating an ElipseX

Suppose a given application needs to supervise and command 10 engines. Each engine must be represented by a drawing on Screen, which displays a green color when operating, and a red color when turned off. It must also display the engine command on Screen, sending instructions for turning it on and off, and its speed must also be displayed.

One possibility is creating an XControl called EngineA, with properties **Status** set to **Boolean** and **Speed** set to **Double**, as seen on the next figures:



Design tab

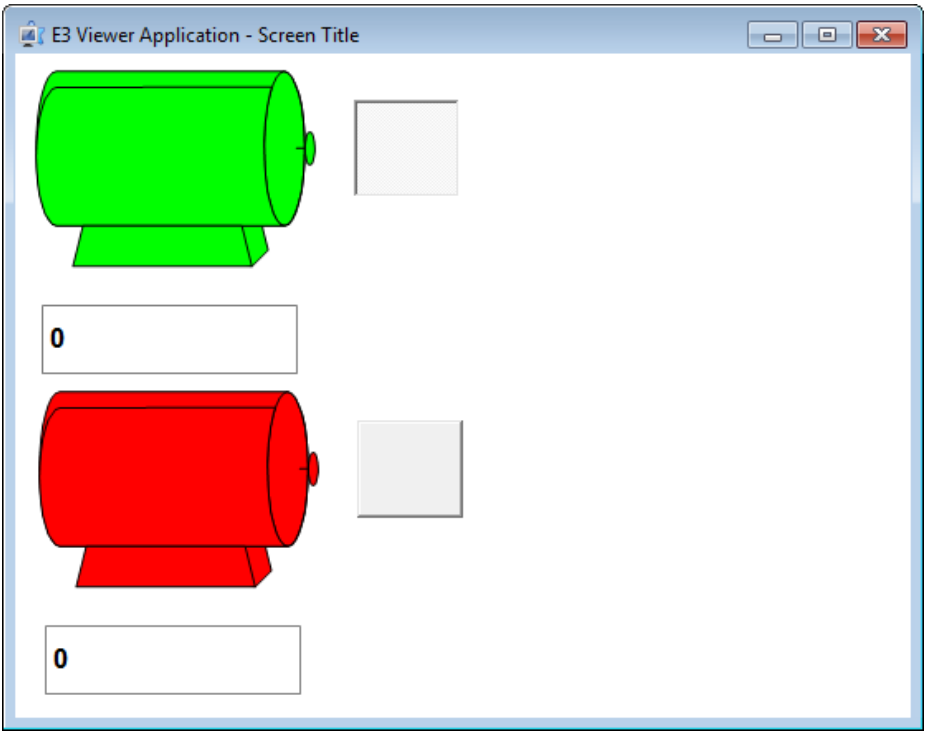


Properties tab

1. To indicate color, engine's **OverrideFillColor** property must be linked to XControl's **Status** property, by using a Digital Link. Set the **OverrideFillMode** property to **2 - SolidFill**.
2. To display speed, Display's **Value** property must be linked to XControl's **Speed** property.
3. The Toggle Button switches the **Status** property value via a Simple Link.

Notice that:

- Links within the library are internal, and their format is **Control_Name.Property_Name**
- The object, after being inserted on a Screen, must have these properties linked to real Tags, for each engine
- A Tag Link to the **Status** property must be performed for each `EngineA`

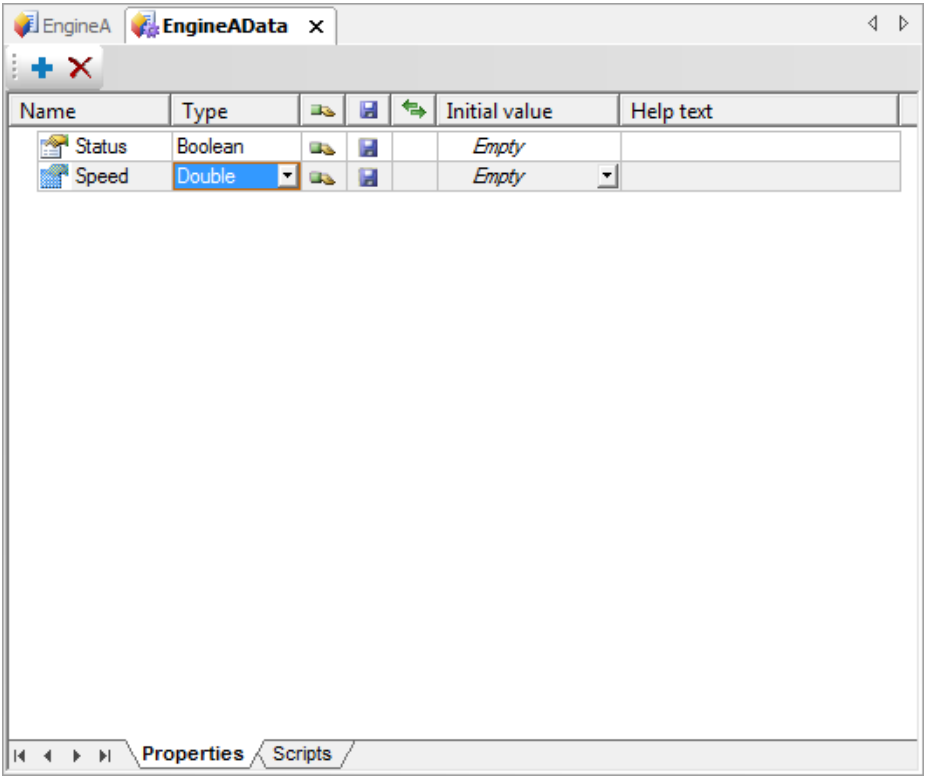


Viewer

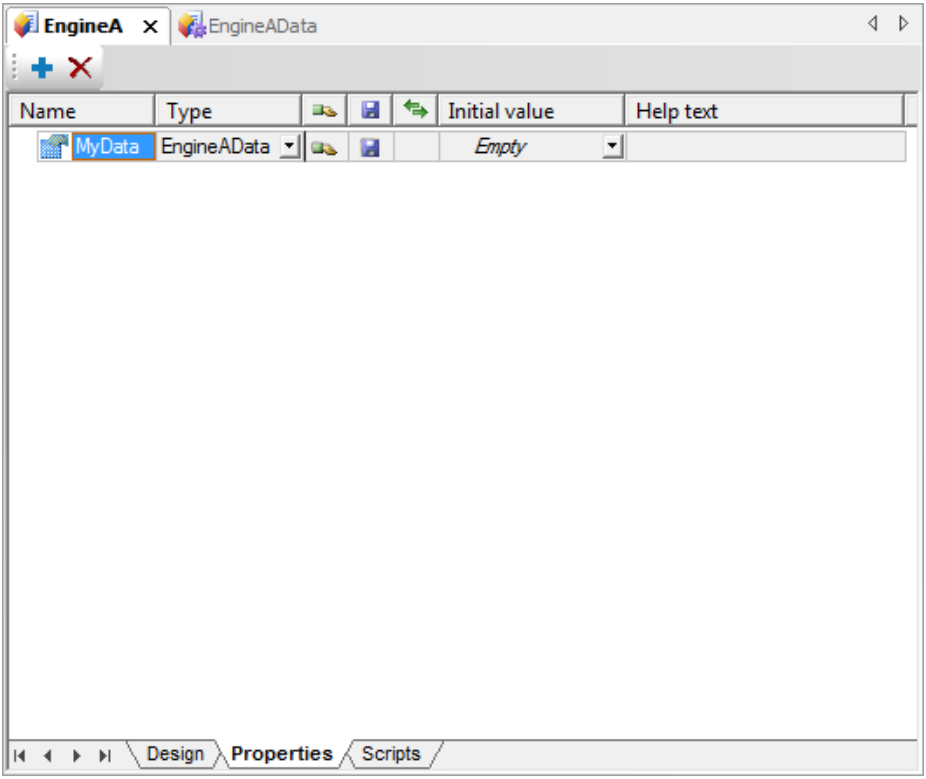
Another broader possibility is to use an XObject for the engine. Thus, all information regarding the engines remains in objects in the server. Hence, it is possible to create several types of interface for the engine (XControls), which brings only relevant information from the server, via XObject.

Then, the EngineA object would have to be modified to point to an XObject, instead of declaring all properties on itself.

1. Create an XObject called EngineAData, and declare **Status** and **Speed** properties in it.
2. Create an XControl EngineA with a single property, called **MyData**, of type **EngineAData**.
3. EngineAData must be inserted in a Data Folder in the server, corresponding to each engine. EngineA, by its turn, will point to the EngineAData needed, with no need to create new Tags.



Configuration of XObject's view



Configuration of XControl's view

4. The **Status** property, linked to engine's **OverrideFillCollor** property, is then **EngineA.MyData.Status**.
5. The **Speed** property, linked to the Display, is then **EngineA.MyData.Speed**.

2.4 Accessing Objects

Following the object oriented programming encapsulation concept, methods and properties are linked to their original objects. This means that users always have to indicate the object from which they are accessing a method or property.

2.4.1 Properties

For references to object properties, users must use the **GetObject** method. Its syntax is the following:

```
Application.GetObject("<object>").<property>
```

Where *<object>* is the name of the object, and *<property>* is the desired property.

Example:

```
Application.GetObject("Data.TempTank2").Type
```

For an easy typing, it is always advisable to use the AppBrowser, which already retrieves the correct syntax.

2.4.1.1 Value Property

In E3, many objects have a common property, called **Value**. In this specific case, users can access this property by using the name of the object itself:

```
Button1 = False
```

Which is equivalent to:

```
Button1.Value = False
```

2.4.2 Methods

The following syntax exemplifies a method call that does not need parameters:

```
Application.GetObject("<object>").<method>
```

Where *<object>* is the specific object, and *<method>* is the desired method.

If the method accepts parameters, use the following syntax:

```
Application.GetObject("<object>").<method>(<parameter>)
```

Where *<parameter>* is the parameter to pass to the method. When there is more than one parameter, use commas to separate them.

If the method returns a result that users want to keep, then the parameters must be placed between parentheses:

```
<V> = Application.GetObject("<object>").<method>(<parameter>)
```

Where *<V>* is the variable that receives the method's result.

2.5 Working with Collections

A collection is an object that manages a set of similar objects. The objects contained in a collection are referred by indexes, similarly to array references. Users can add or remove individual objects from a collection, as seen in the following example:

```
Sub CommandButton1_Click()
  ' Add a pen to the E3Chart1 object
  Screen.Item("E3Chart1").Pens.AddPen "Pen"
End Sub

Sub CommandButton2_Click()
  ' Removes the first pen
```

```
  Screen.Item("E3Chart1").Pens.Remove 0
End Sub
```

NOTE: The first object in a collection has index 1 (one).

All collections have a common property named **Count**, which is the number of existing objects (or children). Example:

```
Sub CommandButton1_Click()
  ' Shows a dialog box with the number of pens
  MsgBox Screen.Item("E3Chart1").Pens.Count
End Sub
```

2.5.1 Accessing Objects using the Item Method

Every collection has an **Item** method, which can be used to access any object within the collection. The **Item** method accepts an *Item* parameter, which can be a number (a positive integer) or the name of the object to access within the collection. The following examples adjust the E3Chart object's second Pen color:

```
Sub CommandButton1_Click()
  ' Changes the color of the third pen
  Screen.Item("E3Chart1").Pens.Item(2).Color = RGB(212, 208, 20)
End Sub
```

Or else:

```
Sub CommandButton1_Click()
  ' Changes the color of a pen named "Pen2"
  Screen.Item("E3Chart1").Pens.Item("Pen2").Color = RGB(212,
208, 20)
End Sub
```

The previous commands are equivalent, the first one indicating the Pen's index within the collection, and the second one indicating its name.

2.6 Set Command

VBScript implements the concept of polymorphism from object-oriented languages, allowing a **Variant**-type variable to assume the form of any object via the **Set** command. This way, a variable works as a pointer to an object, allowing the access to its methods and properties. Example:

```
Sub CommandButton1_Click()
  Set E3Chart = Screen.Item("E3Chart1")
  E3Chart.Pens.Item(2).Color = RGB(212, 208, 20)
End Sub
```

In this example, the same task of the previous example has been accomplished, but the part related to getting the specific object was omitted. Without the **Set**

command, the same call must be written as:

```
Screen.Item("E3Chart1").Pens.Item(2).Color = RGB(212, 208, 20)
```

Apparently, there is no advantage in this case, because users will be able to perform that with a single line of code. But, if they want to perform other operations in the same script right after, the process would be simpler and faster if the call to the **Item** method is not in all lines.

```
Sub CommandButton1_Click()
  ' A bad example:
  Screen.Item("E3Chart1").Pens.Item(0).Color = RGB(212, 208, 20)
  Screen.Item("E3Chart1").Pens.Item(1).Color = RGB(200, 208, 20)
  Screen.Item("E3Chart1").Pens.Item(2).Color = RGB(100, 208, 20)
End Sub

Sub CommandButton1_Click()
  ' A better example:
  Set Pens = Screen.Item("E3Chart1").Pens
  Pens.Item(0).Color = RGB(212, 208, 20)
  Pens.Item(1).Color = RGB(200, 208, 20)
  Pens.Item(2).Color = RGB(100, 208, 20)
End Sub
```

2.7 E3Globals

E3Globals is an E3 module that contains **methods** for global usage. The **GetCurrentWindowsUserName**, **GetLocalTime**, and **GetLocalTimeUTC** methods cannot be used in Links, only in scripts. The rest of the methods can be used in Links, as well as in scripts.

The way of using these methods admits two syntaxes, `E3Globals.<method>` or simply `<method>`, except for the Report object, where the `E3Globals.<method>` syntax is mandatory.

2.7.1 Methods

This section contains information about the methods of the E3Globals module.

2.7.1.1 BShl

BShl(*Value*, *Bits*)

Returns *Value* left-shifted the number of bits specified in *Bits*. This method returns an error in case the *Bits* parameter is outside the allowed range (from 0 to 31). This method is available in Links, as well as in scripts.

2.7.1.2 BShr

BShr(*Value*, *Bits*, *PreserveSign*)

Returns *Value* right-shifted the number of bits specified in *Bits*. This method returns an error in case the *Bits* parameter is outside the allowed range (from 0 to 31). The *PreserveSign* parameter is an optional **Boolean** which, if True, fills in the bits to the left with a copy of the signal bit. The default value of this parameter (False) fills in the bits to the left with zeroes. This method is available in Links, as well as in scripts.

2.7.1.3 Choose

Choose(*Index*, *Values*)

Returns one of the items specified in *Values*, based on *Index* (starts at zero). This method returns Null in case the value of *Index* is less than 0 or greater or equal to the number of values in *Values*. This method is available in Links, as well as in scripts.

NOTE: The **Choose** method does not propagate values' quality nor timestamp. If a Link contains the expression `Choose(TagIndex, Tag1.Value, Tag2.Value, Tag3.Value)`, the result is the chosen value, however with Good (192) quality and the current timestamp. In order to preserve this information, users must specify only the object, as for example `Choose(TagIndex, Tag1, Tag2, Tag3)`.

2.7.1.4 E3Format

E3Format(*Value*, *Format*)

Formats the expression in *Value* using the format specified in *Format*. This format uses the same definitions of the **Format** property of Text, Display, and SetPoint objects. This method is available in Links, as well as in scripts.

2.7.1.5 GetBit

GetBit(*Value*, *BitIndex*)

Returns the value (True or False) of the *Value* bit specified by *BitIndex*. This method returns an error in case the *BitIndex* parameter is outside the allowed range (from 0 to 31). This method is available in Links, as well as in scripts.

2.7.1.6 GetComputerName

GetComputerName()

Returns a **String** containing the name of the local computer. This method is available in Links, as well as in scripts.

2.7.1.7 GetCurrentWindowsUserName

GetCurrentWindowsUserName()

Returns a **String** containing the name of the user logged on the current process. This method is not available in Links, only in scripts.

2.7.1.8 GetLocalTime

GetLocalTime()

Returns the date and time of the local computer, with a precision of milliseconds and in the local time zone. This method is not available in Links, only in scripts.

2.7.1.9 GetLocalTimeUTC

GetLocalTimeUTC()

Returns the date and time of the local computer, with a precision of milliseconds and in the UTC (*Coordinated Universal Time*) time zone. This method is not available in Links, only in scripts.

2.7.1.10 IIf

IIf(*Condition*, *ExprTrue*, *ExprFalse*)

Returns the expression contained in *ExprTrue* if the condition evaluated in *Condition* is true, and the expression contained in *ExprFalse* if the condition evaluated is false. This method is available in Links, as well as in scripts.

NOTE: The **IIf** method does not propagate values' quality nor timestamp. If a link contains the expression `IIf(Tag1.Value == 0, Tag2.Value, Tag3.Value)`, the result is the value of Tag2 or Tag3, however with Good (192) quality and current timestamp. To preserve this information, users must specify only the object, as for example `IIf(Tag1.Value == 0, Tag2, Tag3)`.

2.7.1.11 OPCGetLimit

OPCGetLimit(*Quality*)

Returns the **Limit** information of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts. Possible returned values for this method are:

- **0**: Free
- **1**: Low
- **2**: High

- **3**: Constant

2.7.1.12 OPCGetQuality

OPCGetQuality(*Quality*)

Returns the **Quality** information of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts. Possible returned values for this method are:

- **0**: Bad
- **1**: Uncertain
- **2**: Not used
- **3**: Good

2.7.1.13 OPCGetSubStatus

OPCGetSubStatus(*Quality*)

Returns the **Substatus** information (from 0 to 15) of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts. The OPC Standard specifies the following values:

- **Good Quality**:
  - **0**: non-specific
  - **1**: local override
- **Bad Quality**:
  - **0**: non-specific
  - **1**: configuration error
  - **2**: not connected
  - **3**: device failure
  - **4**: last known value
  - **5**: communication failure
  - **6**: out of service
- **Uncertain Quality**:
  - **0**: non-specific
  - **1**: last usable value
  - **4**: sensor not accurate

- **5**: EU exceeded
- **6**: subnormal

2.7.1.14 OPCGetVendor

OPCGetVendor(*Quality*)

Returns the **Vendor Specific** information (from 0 to 255) of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts.

2.7.1.15 OPCIsBad

OPCIsBad(*Quality*)

Returns True if the OPC Quality is Bad, and False otherwise. This method is available in Links, as well as in scripts.

2.7.1.16 OPCIsGood

OPCIsGood(*Quality*)

Returns True if the OPC Quality is Good, and False otherwise. This method is available in Links, as well as in scripts.

2.7.1.17 OPCIsUncertain

OPCIsUncertain(*Quality*)

Returns True if the OPC Quality is Uncertain, and False otherwise. This method is available in Links, as well as in scripts.

2.7.1.18 OPCMakeQuality

OPCMakeQuality(*QualityFlag*, *SubStatus*, *Limit*, *Vendor*)

Returns a new OPC Quality value, built using the values passed in the *QualityFlag*, *SubStatus*, *Limit*, and *Vendor* parameters. This method is available in Links, as well as in scripts. Possible values for each one of the parameters are:

- **QualityFlag**: Specifies the quality of the value
  - **0**: Bad
  - **1**: Uncertain
  - **3**: Good
- **SubStatus**: Specifies the substatus of the value (between 0 and 15, see the **OPCGetSubStatus** method for possible values). If this parameter is omitted, assumes 0 (zero)

- **Limit**: Specifies the limit of the value. If this parameter is omitted, assumes 0 (zero)
  - **0**: Free
  - **1**: Low
  - **2**: High
  - **3**: Constant
- **Vendor**: Specific value of the vendor (between 0 and 255). If this parameter is omitted, assumes 0 (zero)

2.7.1.19 OPCSetLimit

OPCSetLimit(*Quality*, *Limit*)

Changes the **Limit** information of an OPC Quality and returns the modified value. This method is available in Links, as well as in scripts. Possible values for the *Limit* parameter are:

- **0**: Free
- **1**: Low
- **2**: High
- **3**: Constant

2.7.1.20 OPCSetQuality

OPCSetQuality(*Quality*, *QualityFlag*)

Changes the **Quality** information of an OPC Quality and returns the modified value. This method is available in Links as well as in scripts. Possible values for the *QualityFlag* parameter are:

- **0**: Bad
- **1**: Uncertain
- **2**: Not used
- **3**: Good

2.7.1.21 OPCSetSubStatus

OPCSetSubStatus(*Quality*, *SubStatus*)

Changes the **Substatus** information (from 0 to 15) of an OPC Quality and returns the modified value. This method is available in Links, as well as in scripts. The OPC Standard specifies the following values:

- **Good Quality**:
  - **0**: non-specific
  - **1**: local override
- **Bad Quality**:
  - **0**: non-specific
  - **1**: configuration error
  - **2**: not connected
  - **3**: device failure
  - **4**: last known value
  - **5**: communication failure
  - **6**: out of service
- **Uncertain Quality**:
  - **0**: non-specific
  - **1**: last usable value
  - **4**: sensor not accurate
  - **5**: EU exceeded
  - **6**: subnormal

2.7.1.22 OPCSetVendor

OPCSetVendor(*Quality*, *Vendor*)

Changes the **Vendor Specific** information (from 0 to 255) of an OPC Quality and returns the modified value. This method is available in Links, as well as in scripts.

2.7.1.23 SetBit

SetBit(*Value*, *BitIndex*, *BitValue*)

Sets the value of *Value* (True or False) of the bit specified by *BitIndex* (from 0 to 31) to *BitValue*. This method returns an error in case the *BitIndex* parameter is outside the allowed range (between 0 and 31). This method is available in Links, as well as in scripts.

2.7.1.24 SourceTypeName

SourceTypeName(*SourceType*)

Returns a **String** with the description of the active Measurement Source (the **ActiveSource** property of Analog Measurement and Discrete Measurement objects in Elipse Power). This method is available in Links, as well as in scripts. Possible values for *SourceType* parameter are the following:

- **-1**: Empty **String**
- **0**: Active Source
- **1**: SCADA
- **2**: Operator
- **3**: Control Center
- **4**: Billing
- **5**: Calculated
- **6**: Database
- **100**: Topology Processor
- **101**: Power Flow
- **102**: State Estimator
- **103**: Load Shedding

NOTE: In case the value passed on *SourceType* parameter is not one of the possible values, this method returns "???".

2.8 General Events, Methods, and Properties of Objects

This section contains information about general **events**, **methods**, and **properties** of objects.

2.8.1 Events

Events are occurrences related to an object, which allow triggering scheduled actions. Basically, there are two types of events: **Physical** (or **External**) and **Internal**. **Physical Events** are, for example, user actions. In case users type on the keyboard, the relevant information can be the pressed key or, if users point and click the mouse, the relevant information can be the pointer position and the button status. **Internal Events** are, for example, value changes in a variable (Tag) in an application.

Since the Tag can be linked to an external device, it is possible to say that internal events can have physical associations, such as temperature changes in a chamber, for example.

2.8.1.1 Event Variables

Event Variables are created when the event starts. To be used, they must be linked to parameters in the event's script call.

The following example is a procedure call linked to the **KeyDown** event of `AnObject`.

```
Sub AnObject_KeyDown(KeyCode, Shift)
```

Notice that in this call there are two variables, *KeyCode* and *Shift*. E3 will automatically assign values to these variables when this event occurs. In this case, *KeyCode* will receive the code of the pressed key, and *Shift* will be True or False, whether the SHIFT key is pressed or not.

2.8.1.2 OnStartRunning

OnStartRunning()

Occurs as soon as an object starts. Example (`Months` is an Internal Tag and uses the **OnStartRunning** event to initialize an array):

```
Sub Months_OnStartRunning()
  Value = Array("January", "February", "March", "April",_
    "May", "June", "July", "August", "September", "October",_
    "November", "December")
End Sub
```

NOTE: To access this array, users must copy the **Value** property to a local variable.

2.8.1.3 OnStopRunning

OnStopRunning()

Occurs when the execution of an instance of this object finishes. Use the **OnStopRunning** event to perform termination operations for the object. Example:

```
Sub InternalTag1_OnStopRunning()
  ' When InternalTag1 object finishes
  ' it assigns False to InternalTag2
  Set tag2 = Application.GetObject("Data.InternalTag2")
  tag2.Value = False
End Sub
```

2.8.2 Methods

This topic lists common methods of E3 objects. Each entry displays the name of the method with its respective parameters, in its correct syntax, and also a usage example.

2.8.2.1 Calling Methods

Many predefined methods have parameters, which can (or should) be passed at method's call. For this, VBScript has a rule that must be followed. If the method is used in an assignment, its parameters must be enclosed in brackets. For example, check this **GetObject** method's call:

```
obj = Application.GetObject("data.tag001")
```

If a method is called alone, then users must remove the brackets. For example, check this **SetVariableValue** method's call:

```
Screen.Item("Query").SetVariableValue Value, 12
```

Brackets used when citing methods in this manual are only an indication to differentiate them from properties. In scripts, this rule must be followed.

2.8.2.2 Activate

Activate()

Activates a currently inactive object. Example:

```
Sub CommandButton1_Click()
  Dim obj, tag
  Set obj = Application.GetObject("Data")
  ' Create a new object and let it disabled (False).
  Set tag = obj.AddObject("DemoTag", False)
  ' Setup the new object parameters.
  tag.Name = "tag001"
  tag.Type = 3
  ' Enable the object (start it up).
  tag.Activate()
End Sub
```

2.8.2.3 AddObject

AddObject(*ClassName*[, *Activate*[, *ObjectName*]])

The **AddObject** method adds a new object in the application. This method has the *ClassName* parameter, which indicates the type of the object to create. For example, to create a rectangle on a Screen, the *ClassName* parameter must be "DrawRect". The created object is contained inside the object that called the **AddObject** method, and can be accessed using the **Item** method.

The *Activate* parameter is optional and indicates whether the object is enabled after creating it or not. If the object is activated, Links and scripts remain enabled. If the object is created with *Activate* equal to False, it can be later activated using the **Activate** method.

The *ObjectName* parameter is also optional and indicates a name for the new object. In case this name already exists, the new name is automatically incremented. If this parameter is not informed, the new object is named after the class defined in the *ClassName* parameter.

The object is created only if its type is compatible with the object that contains it. To ensure that the object was created, the **IsObject** method can be used.

NOTE: Only objects that have the **Insert** menu option can use this method.

2.8.2.4 Context

Context(*ContextName*)

Returns the object implementing the context indicated by the parameter *ContextName*, which must be a **String** surrounded by quotation marks. This method fails if no object on the upper hierarchy of the object calling this method implements this context. The following contexts are available:

- **Container**: Server and Viewer objects (objects inserted in project files, or in folders inside projects)
- **Area**: Alarm Areas, or any server object whose property **IsAlarmArea** is set to True

NOTE: Context names are case-insensitive. To check the context to which an object belongs, just open its Properties window, select the **Item** tab, and then check its valid contexts in the **Contexts** frame. In case an object defines more than one context, their names are presented in alphabetical order, separated by commas.

2.8.2.5 Deactivate

Deactivate()

Deactivates a previously created object, or an object previously activated using the **Activate** method. Users can deactivate an object when it is necessary to perform a previous configuration (property initialization, for example), or when they want to perform tests in which it is necessary that the object is neither present nor activated. Example:

```
Sub CommandButton1_Click()
  Dim obj, newObj
  Set obj = Application.GetObject("Data")
  Set newObj = obj.AddObject("DemoTag", True)
  ' Deactivate the object.
  newObj.Deactivate()
```

```
End Sub
```

2.8.2.6 DeleteObject

DeleteObject(*ChildName*)

Erases the specified object from the project. The *ChildName* parameter is a **String** (ignores capitalization) and indicates the child object to delete. This method returns True if the object was successfully deleted, or False, if the child object does not exist.

To delete an object using a reference to an element, the parent object's **DeleteObject** method is used. Example:

```
Sub CommandButton1_Click()
  Set obj = Application.GetObject("Data")
  If obj.DeleteObject("Tag001") Then
    MsgBox("Tag successfully deleted!")
  Else
    MsgBox("Deleting failed: tag does not exist.")
  End If
End Sub
```

NOTE: Only objects that have the **Insert** menu option can use this method.

2.8.2.7 GetChildObject

GetChildObject(*ObjectPath*)

The **GetChildObject** method returns a reference to the child object pointed by the *ObjectPath* parameter. Then it is possible to access all properties and methods of this object, similar to the **GetObject** method. This method fails if the path pointed by the *ObjectPath* parameter contains a property or method at its end. The path pointed by the child object is not a path starting at root (the .prj file) but instead a path starting at the object where this method is called.

NOTE: This method **DOES NOT** exist in the server's Application object nor in the Application Folders, however it exists in the Viewer's Application object, and it is accessible even in a **Read-Only** Viewer.

2.8.2.8 GetObject

GetObject(*ObjectPath*)

This method returns a reference to the object specified by *ObjectPath*. This allows accessing all object's properties and methods. This is a common practice in E3 scripts programming. It facilitates handling objects and makes code more intelligible. Example:

```
Sub CommandButton1_Click()
  ' Assign the value 20 to InternalTag1's Value property
```

```
  ' which is inside Data.
  Set tag = Application.GetObject("Data.InternalTag1")
  tag.Value = 20
End Sub
```

2.8.2.9 Item

Item(*ItemId*)

Returns a reference to the child object *ItemId* of the object that called this method. The **Item** method can search for an object either by its name or by its index (an integer starting at 1 and less or equal to the **Count** property). If the specified index or name is valid, the **Item** method returns the object's reference. Otherwise it will return an "Invalid Parameter" error. Example:

```
Sub Screen1_Click()
  ' Assigns to obj the reference to the child object Button1
  ' of Screen1.
  Set obj = Item("Button1")
  ' Set the obj's BackColor property, that is,
  ' Button1.
  obj.BackColor = RGB(255, 0, 0)
End Sub
```

2.8.2.10 Save

Save()

This method saves the specified object, which was modified at run time. Children objects will also be saved, according to the parent object's specifications. This method is not valid for Screen and Viewer objects. Example:

```
Sub CommandButton1_Click()
  Set area = Application.GetObject("ConfigAlarms")._
    AddObject("Area", true)
  Application.GetObject("ConfigAlarms").Save()
End Sub
```

NOTE: Changes made at run time and saved in the object are only visible in Studio after updating the project, which can be performed by right-clicking the project's name and then selecting the **Refresh** option.

2.8.3 Properties

Each object has **Properties**, which are used to save information about its characteristics. For example, a **Rectangle**-type object has a **Name** property, which stores the object's name, and the **Width** and **Height** properties, which store its width and height, respectively, among other properties.

This chapter lists all properties common to E3 objects. Each entry has the name of the property, its description and, when applicable, an example of its usage.

Properties are identified by an icon indicating the data type supported by their content. The available data types are the following:

Available data types

ICON	DATA TYPE	DESCRIPTION
	Boolean	Returns True or False.
	Numeric	Returns an integer or double (positive or negative), as defined by the property.
	Date	Returns a date in the Gregorian calendar (beginning in 1899).
	Text	Returns a text.
	Variant	Returns a variable type, which can assume several formats.
	Color	Returns a color in RGB format.
	Link	Returns a Link between objects.
	Enum	Returns a determined set of values.

Some properties can propagate their values to the same property in their child objects. In this case, they are called propagatable properties. Users can, however, force a child object's property to behave distinctively.

NOTE: E3 uses the Himetric system to define coordinates and widths. In this system, each logical unit is equivalent to a thousandth centimeter; that is, 1,000 units is equivalent to one centimeter. Thus, this is the standard adopted when describing E3 properties, when applicable.

2.8.3.1 Application

The **Application** property returns the Application object related to the current object's context. With this Application object it is possible, for example, to search for other objects in the application. Example:

```
Sub Screen1_Click()
  ' When clicking the screen, sets a value
  ' and displays a message box
  set obj = Application.GetObject("Data.InternalTag1")
  obj.Value = 100
  MsgBox "Value of InternalTag1: " & obj.Value
End Sub
```

2.8.3.2 Count

9 Returns the number of child objects(items) an object contains. This property works together with the **Item** method. If the object does not have any child objects, the value of this property is 0 (zero). Example:

```
Sub Screen1_Click()
  ' Searches all Screen objects and sets
  ' the ForegroundColor property to red
  Dim obj
  For i = 1 To Count
    Set obj = Item(i) ' Gets a child object
    obj.ForegroundColor = RGB(255, 0, 0)
  Next
End Sub
```

2.8.3.3 DocString

A Free text that enables documenting object's functionality or features by the project's programmers. Example:

```
Sub CommandButton1_Click()
  Docstring = "This button activates the system condenser."
  MsgBox Docstring
End Sub
```

2.8.3.4 Links

Returns an object that is a collection of connections (or Links) of any E3 object. This property is only accessible at run time. Check the item **Collection of Links** for more information about the collection of objects returned by this property.

2.8.3.5 Name

A Identifies every object present in the application. Modifying this property implies in modifying all other properties or scripts that use this object. Hence, it is not advisable to change this property at run time. Example:

```
Sub CommandButton9_Click()
  MsgBox "Screen's name is " & (Screen.Name)
End Sub
```

2.8.3.6 Parent

Returns the object's parent object. So, if an object is inserted on a Screen, then the **Parent** property returns "Screen". Likewise, if an Internal Tag is inserted directly under a Data Server, InternalTag's **Parent** property then points to the Data Server. Example:

```
Sub Rectangle1_Click()
  ' When clicking Rectangle1,
```

```
  ' changes the foreground color of Rectangle2
  Parent.Item("Rectangle2").ForegroundColor = RGB(255, 0, 0)
End Sub
```

2.8.3.7 PathContainer

A Returns a **String** with the path of the object containing the current object, including its Folders. This value is determined only at request, thus it is not recommended to create Links to this property.

2.8.3.8 PathName

A Identifies a path name in the application. This property is only available at run time. Example:

```
Sub CommandButton9_Click()
  MsgBox "Screen path is " & (Screen.PathName)
End Sub
```

2.8.3.9 PathVolume

A Returns a **String** with the name of the .prj or .lib file containing the object. In Studio, this property returns the project's or library's full path (c:\folder\folder \volume.prj). At run time, the objects running in the Viewer always return an empty **String**. However, objects running in the server return the project's or library's relative path, according to the way it is stored in the Domain (volume.prj). This value is determined only at request, thus it is not recommended to create Links to this property.

2.8.4 Collection of Links

This section contains information about **methods** and **properties** common to the collection of connections (or Links) of any E3 object, returned by the **Links** property.

2.8.4.1 Common Methods

This section contains information about methods common to the collection of Links returned by the **Links** property.

2.8.4.1.1 CreateLink

CreateLink(*Property*, *Source*[, *Type*])

This method allows creating a Link with an object's property. In case of success, this method returns the object just created. Otherwise, a script error occurs and this method returns `Nothing`. This method has the following parameters:

- ***Property***: specifies the name of the property to which the Link will be created
- ***Source***: specifies the name of Link's source object

- ***Type*** (optional): specifies the type of Link to create. When omitted, creates a Simple Link

NOTE: Not all existing properties in an object allow creating Links. To check which properties allow this feature, go to the **Links** tab. If the property is invalid for a Link, it does not exist, or it already has a Link, then a script error occurs.

Available options for Type parameter

OPTION	DESCRIPTION
0 - Simple Link	In a Simple Link, its source value is copied to the property every time it is modified.
1 - Bidirectional Link	A Bidirectional Link works as a Simple Link, but when there is a variation in the property, its value is copied to the source, thus generating a two-direction Link.
2 - Analog Link	An Analog Link specifies a conversion scale between the source variable and the property.
3 - Digital Link	In a Digital Link, fixed or alternating values are set to the property, and these values are set according to whether the source is True or False.
4 - Table Link	In a Table Link, users can specify conditions between the variables, the values, and the destination. On the table users can specify minimum and maximum values, and other configurations.
5 - Reverse Link	A Reverse Link is a unidirectional Link from the property to the source.
6 - Multisource Link	A Multisource Link is similar to a Table Link, except that each Link row allows retrieving its value from a different source.

Example:

```
Sub CommandButton1_Click()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("Text1").Links.Item("Value")
  If Bind Is Nothing Then
    MsgBox "Text1 is not linked to any object."
    Dim Source
    Source = "Data.InternalTag1.Value"
```

```
    MsgBox "Creating a link in '" & Source & "'."
    Set Bind = Screen.Item("Text1").Links._
      CreateLink("Value", Source, 0)
    Bind.BiDirectional = Screen.Item("BiDirectional").Value
    Bind.Reverse = Screen.Item("Reverse").Value
    MsgBox "Type: " & TypeName(Bind)
  Else
    MsgBox "Text1 is already linked to '" &Bind.Source& "'."
  End If
End Sub
```

2.8.4.1.2 Item

Item(*Property*, *Index*)

This method returns a Link object from a specific property of an object. If it is a text, *Property* specifies the name of the property to access. The Link can also be numerically accessed by *Index*. This index must be inside the range 1 up to **Count**. In case there is no Link to the property, or the index is invalid, then a script error occurs. As in other collections, The Links collection allow using VBScript's **For Each** command. Example:

```
Sub Texto1_Click()
  For Each Link In Links
    MsgBox "Link source: " & Link.Source
  Next
End Sub
```

2.8.4.1.3 RemoveLink

RemoveLink(*Property*)

This method removes a Link from a property specified by *Property*, if it exists. If a Link to the specified property does not exist, this method has no effect. Example:

```
Sub CommandButton2_Click()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  If Bind Is Nothing Then
    ' If the link does not exist
    MsgBox "ScrollBar1 is not linked."
  Else
    MsgBox "ScrollBar1 is linked to '" & Bind.Source & "'"
    MsgBox "Removing the link."
    Screen.Item("ScrollBar1").Links.RemoveLink("Value")
  End If
End Sub
```

2.8.4.2 Common Properties

This section contains information about properties common to the collection of Links returned by the **Links** property.

2.8.4.2.1 Count

9 Returns the number of child objects (items) of a Link collection. This property works together with the **Item** method. If the collection does not have children, the returned value is 0 (zero).

2.8.4.3 Links

This section contains information about **Link**-type objects inside the collection of Links returned by the **Links** property. The available types of Links are the following:

- **Simple**
- **Bidirectional**
- **Analog**
- **Digital**
- **Table**
- **Reverse**
- **Multisource**

2.8.4.3.1 Common Properties

This section contains information about the properties common to objects contained in the collection of Links returned by the **Links** property.

2.8.4.3.1.1 Property

A Specifies the name of the property it is linked to. When modified, moves the Link to another property of the same object. Example:

```
Sub CommandButton1_Click()
  Dim bind
  Set bind = Screen.Item("TableBind").Links.Item(1)
  bind.Property = "Caption"
End Sub
```

2.8.4.3.1.2 Source

A Specifies Link's source, which can be the name of another application object, or a more complex expression accessing several objects. Example:

```
Sub CommandButton25_Click()
  Dim bind
  Set bind = Screen.Item("TableBind").Links.Item(1)
  bind.Source = "Data.DemoTag1.Value"
End Sub
```

2.8.4.3.1.3 Type

This is a read-only property, and informs the Link's type. The available options are:

Available options for Type

OPTION	DESCRIPTION
0 - bsSimple	Simple Link.
1 - bsSimpleBiDir	Bidirectional Link.
2 - bsAnalog	Analog Link.
3 - bsAnimation	Digital Link.
4 - bsTable	Table Link.
5 - bsReverse	Reverse Link.
6 - bsMultiSource	Multisource Link.

2.8.4.3.2 Simple Link

The Simple Link object does not have events, methods, or properties associated to it.

2.8.4.3.3 Bidirectional Link

This section contains information about **properties** of a Bidirectional Link object. This object does not have associated events nor methods.

2.8.4.3.3.1 Properties

This section contains information about properties of a Bidirectional Link object.

BiDirectional

True when a Link is Bidirectional, False when a Link is Simple or Reverse.

2.8.4.3.4 Analog Link

This section contains information about **properties** of an Analog Link object. This object does not have associated events nor methods.

2.8.4.3.4.1 Properties

This section contains information about properties of an Analog Link object.

DstHiValue

Specifies the highest value reached by the property. Example:

```
Sub DstHiValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar1").Max = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing DstHiValue from " &_
      Bind.DstHiValue & " to " & Value
    Bind.DstHiValue = Value
  End If
End Sub
```

DstLoValue

Specifies the lowest value reached by the property. Example:

```
Sub DstLoValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar1").Min = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing DstLoValue from " &_
      Bind.DstLoValue & " to " & Value
    Bind.DstLoValue = Value
  End If
End Sub
```

SrcHiValue

Specifies the highest value reached by the source. Example:

```
Sub SrcHiValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar2").Max = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing SrcHiValue from " &_
      Bind.SrcHiValue & " to " & Value
    Bind.SrcHiValue = Value
```

```
  End If
End Sub
```

SrcLoValue

9 Specifies the lowest value reached by the source. Example:

```
Sub SrcLoValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar2").Min = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing SrcLoValue from " &_
      Bind.SrcLoValue & " to " & Value
    Bind.SrcLoValue = Value
  End If
End Sub
```

NOTE: In case the values specified for **SrcHiValue** and **SrcLoValue** properties are the same, there will be no scale, then the Link will work as a Simple Link.

2.8.4.3.5 Digital Link

This section contains information about **properties** of a Digital Link object. This object does not have associated events nor methods.

2.8.4.3.5.1 Properties

This section contains information about properties of a Digital Link object.

BlinkOff

◩ When this property is set to True, the connected property alternates periodically between values of **OffValue** and **BlinkOffValue** properties, in case the source returns False. Example:

```
Sub BlinkOff_Change()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("Rectangle1")._
    Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no link."
  Else
    MsgBox "Rectangle1 is linked to " & Bind.Source & "'"
    MsgBox "Changing BlinkOff from " &_
      Bind.BlinkOff & " to " & Value
    Bind.BlinkOff = Value
```

```
  End If
End Sub
```

BlinkOffValue

Specifies an alternative value the property assumes periodically when the source's expression returns False, and the **BlinkOff** property is set to True. Example:

```
Sub BlinkOffValue_Click()
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to "& Bind.Source & "'"
      MsgBox "Changing BlinkOffValue from " &_
        Bind.BlinkOffValue & " to " & Value
      Bind.BlinkOffValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

BlinkOn

When this property is set to True, the connected property alternates periodically between values of **OnValue** and **BlinkOnValue** properties, in case the source returns True. Example:

```
Sub BlinkOn_Change()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("Rectangle1")._
    Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no link."
  Else
    MsgBox "Rectangle1 is linked to " & Bind.Source & "'"
    MsgBox "Changing BlinkOn from " &_
      Bind.BlinkOn & " to "_ & Value
    Bind.BlinkOn = Value
  End If
End Sub
```

BlinkOnValue

Specifies an alternative value the property assumes periodically when the source's expression returns True, and the **BlinkOn** property is set to True. Example:

```
Sub BlinkOnValue_Click()
```

```
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to " & Bind.Source & "'"
      MsgBox "Changing BlinkOnValue from " &_
        Bind.BlinkOnValue & " to " & Value
      Bind.BlinkOnValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

OffValue

Specifies the value a property assumes when the source's expression returns False. Example:

```
Sub OffValue_Click()
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to "& Bind.Source & "'"
      MsgBox "Changing OffValue from " &_
        Bind.OffValue & " to " & Value
      Bind.OffValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

OnValue

Specifies the value a property assumes when the source's expression returns True. Example:

```
Sub OnValue_Click()
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
```

```
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to "& Bind.Source & " "
      MsgBox "Changing OnValue from " &_
        Bind.OnValue & " to " & Value
      Bind.OnValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

2.8.4.3.6 Table Link

This section contains information about **methods** and **properties** of a Table Link object. This object does not have associated events.

2.8.4.3.6.1 Methods

This section contains information about methods of a Table Link object.

InsertRow

InsertRow([*Row*])

Inserts a new row on the table. The *Row* parameter is optional and specifies at which table position the row should be inserted. When omitted, it assumes the standard behavior of inserting a row at the end of the table, which is the same as using *Row* equal to -1. When it is informed and is not equal to -1, it must be a value between 1 and **Count**, and the new row moves the other rows greater or equal to the index towards index ascending direction. A new row always assumes the following standard values for its properties:

- **Min**: 0.0
- **Max**: 1.0
- **Blink**: False
- **BlinkValue**: 0.0
- **Value**: 0.0

Example:

```
Sub Rectangle1_Click()
  On Error Resume Next
  Dim Bind
  Set Bind =
```

```
Screen.Item("Rectangle1").Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no links."
  Else
    Dim row
    row = Screen.Item("SetPointRow").Value
    MsgBox Bind.RowCount
    If (row < 1 Or row > Bind.RowCount) Then
      MsgBox "Invalid row number: " & row
    Else
      MsgBox "Adding a row at: " & row
      Bind.InsertRow(row)
      If row = -1 Then
        row = Bind.RowCount
        Bind.Value(line) =
Screen.Item("RectangleValue").ForegroundColor
        Bind.BlinkValue(line) =
Screen.Item("RectangleBlinkValue").ForegroundColor
        Bind.Max(line) = Screen.Item("SetPointMax").Value
        Bind.Min(line) = Screen.Item("SetPointMin").Value
        Bind.Blink(line) = Screen.Item("CheckBoxBlink").Value
      End If
    End If
  End If
End Sub
```

Item

Item(*ItemId*)

Returns a reference to a Table Link Row object, indicated by *ItemId*.

RemoveRow

RemoveRow(*Row*)

Removes a row at a specified index. The *Row* parameter determines the table row to remove (it must be a value between 1 and **Count**). Example:

```
Sub RemoveRow_Click()
  On Error Resume Next
  Dim Bind
  Set Bind =_
    Screen.Item("Rectangle1").Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no link."
  Else
    MsgBox "Rectangle1 is linked to '" & Bind.Source & "'"
    Dim row
    row = Screen.Item("Row").Value
    MsgBox "Removing line " & row
    Bind.RemoveRow row
  End If
```

```
End Sub
```

2.8.4.3.6.2 Properties

This section contains information about properties of a Table Link object.

Count

9 Informs the number of rows on a table. This is a read-only property.

2.8.4.3.6.3 Table Link Row

This section contains information about **properties** of a Table Link Row object. This object does not have associated events nor methods.

Propriedades

This section contains information about properties of a Table Link Row object.

Blink

◩ Determines that when the source is within this Row's interval, the property alternates periodically between values established in the **Value** and **BlinkValue** properties. Example:

```
Sub CheckBox1_Click()
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(1).Blink = Value
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(2).Blink = Value
End Sub
```

BlinkValue

∨ Specifies the alternative (blinking) value the property assumes when source is within the specified Row's interval, and the **Blink** property is set to True. Example:

```
Sub CommandButton1_Click()
  Dim Color
  ' Picks a color
  Application.ShowPickColor Color, 0, 100, 100
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(1).BlinkValue = Color
End Sub
```

Max

9 Specifies the highest source value for a Table Row.

Min

9 Specifies the lowest source value for a Table Row. Example (applicable for both **Max** and **Min** properties):

```
Sub CommandButton1_Click()
  Set Bind = _
```

```
    Screen.Item("Rectangle1").Links.Item("ForegroundColor")
  Set Line1 = Bind.Item(1)
  Line1.Min = 0
  Line1.Max = 20
  Set Line2 = Bind.Item(2)
  Line2.Min = 21
  Line2.Max = 100
End Sub
```

Value

Specifies the value the property assumes when the source is within the specified Row's interval. Example:

```
Sub CommandButton1_Click()
  Dim Color
  ' Picks a color
  Application.ShowPickColor Color, 0, 100, 100
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(1).Value = Color
End Sub
```

2.8.4.3.7 Reverse Link

This section contains information about **properties** of a Reverse Link object. This object does not have associated events nor methods.

2.8.4.3.7.1 Properties

This section contains information about properties of a Reverse Link object.

Reverse

True when a Link is Reverse, False when a Link is Bidirectional or Simple.

2.8.4.3.8 Multisource Link

This section contains information about **methods** and **properties** of a Multisource Link object. This object does not have associated events.

2.8.4.3.8.1 Methods

This section contains information about methods of a Multisource Link object.

InsertRow

InsertRow(*InsertAtRow*)

Inserts a new Row on the Multisource Link table.

Item

Item(*ItemId*)

Returns a reference to a Multisource Link Row object, indicated by *ItemId*.

RemoveRow

RemoveRow(*Row*)

Removes the Row at the index indicated by the *Row* parameter.

2.8.4.3.8.2 Properties

This section contains information about properties of a Multisource Link object.

AdviseAll

◩ This property keeps all Links of the Multisource Link table in **Advise** (active) mode. The default value of this property is True. For applications created in previous versions, this property is set to False, for compatibility reasons.

Count

9 This property returns the number of Rows on the Multisource Link table.

2.8.4.3.8.3 Multisource Link Row

This section contains information about **properties** of a Multisource Link Row object. This object does not have associated events nor methods.

Propriedades

This section contains information about properties of a Multisource Link Row object.

Max

9 Maximum interval for a Link value indicated by the **Source** property.

Min

9 Minimum interval for a Link value indicated by the **Source** property.

Source

A Specifies Multisource Link Row's source, which can be the name of another application object, or a more complex expression accessing several objects.

CHAPTER 3

User Libraries

This section contains information about XControls and XObjects, and also ElipseX Properties.

3.1 XControls and XObjects

This section contains information about **events** and **properties** of XControls and XObjects. These objects do not have methods associated to them.

3.1.1 Events

This section contains information about the events of XControls and XObjects.

3.1.1.1 Constructor

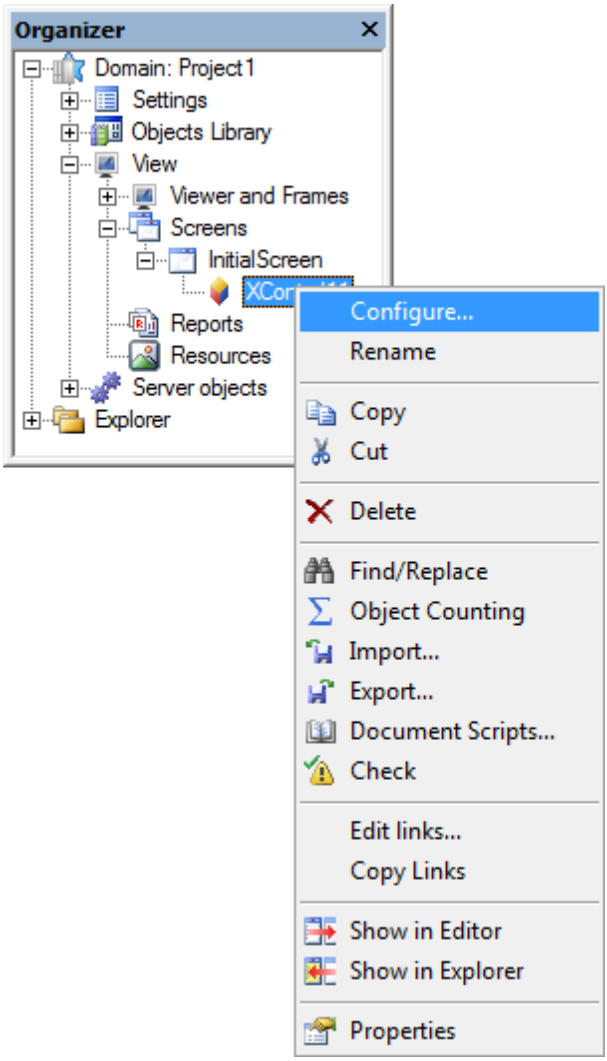
Constructor()

Triggered when the ElipseX is started. Users can use this event to run a script that sets internal values of an ElipseX, for example.

3.1.1.2 CustomConfig

CustomConfig()

Allows automated configurations on ElipseX instances. A configuration option appears on a contextual menu of an ElipseX instance when there is a script linked to the **CustomConfig** event of the ElipseX definition. When this option is selected on this menu, that event is triggered. The text that appears on this menu option can be set on the **CustomConfigText** property of the ElipseX definition. If this property is empty, the displayed text is "Configure".



Contextual menu of an ElipseX instance

NOTE: A script linked to the **CustomConfig** event runs on Studio, where objects are not active. Therefore, its behavior is different from the usual.

3.1.2 Properties

This section contains information about the properties of XControls and XObjects.

3.1.2.1 CustomConfigText

A Indicates a text that appears on the contextual menu for the configuration option of the ElipseX instance. This option only appears if there is a script associated to the **CustomConfig** event of the ElipseX definition. If the property value is empty, the text "Configure" is then displayed in the menu option. The default value of this property is is an empty **String**.

3.2 ElipseX Properties

This section contains information about **events** and **properties** of the Property object of an ElipseX. This object does not have methods associated to it.

3.2.1 Events

This section contains information about the events of the Property object of an ElipseX.

3.2.1.1 OnPropertyChanged

OnPropertyChanged()

It is triggered when an ElipseX property is modified. You may use this event to trigger scripts that execute actions according to a certain ElipseX status.

3.2.2 Properties

This section contains information about the properties of the Property object of an ElipseX.

3.2.2.1 DefaultValue

v Defines the Property's initial value when a new instance of the object is created. If the Property is configured as **retentive**, then it is initialized with this value every time the object is loaded. The default value of this property is Empty.

3.2.2.2 HelpString

A Text containing the Property's description. This text is displayed at the bottom of the Property List in Studio when the Property is selected. The default value of this property is an empty **String**.

3.2.2.3 Persistable

◩ Indicates whether the Property is saved in the project file (True) or is available only at run time (False). The default value of this property is True. If this property is

configured as False, the Property cannot be edited in Studio anymore, saved nor read from the project file. However, the Property is still visible in the AppBrowser. This property is represented by the icon .

When this property is configured as True, the Property receives its default value (the **DefaultValue** property) only when creating an instance. If the **DefaultValue** property changes, the object instances already created are not affected.

When this property is configured as False, the Property receives its default value whenever an instance is loaded, that is, whenever the **DefaultValue** property changes, all instances already created are initialized with the new default value.

3.2.2.4 Public

When an ElipseX Property is public (True), it is visible outside the Library. Otherwise, the Property is internal and only visible to the object. The default value of this property is True. A public Property is represented by the icon .

3.2.2.5 Retentive

Indicates if the current Property's value at run time is persisted in the Domain file (True), while the Domain is loaded. Retentive Properties have the following behavior:

- Propagates its value to the server in Standby
- Keeps its value if the application is updated at run time
- Keeps its value if the application is stopped (as long as the Domain is not unloaded)

NOTES:
- This property is only available for Properties whose data type is not an object (**Variant**, **Double**, **Integer**, etc.)
- Using this property in True implies in raising E3Run's memory and CPU usage, therefore it must be used carefully

3.2.2.6 Type

A Determines the type of values accepted by a Property (for example, **Boolean**, **Double**, **Integer**, **Variant**, etc.). When specifying an object type (as for example DemoTag, IOTag, XObject, etc.), this property has the following behavior:

- **In case the ElipseX is inactive**: This property works as a **String**, which specifies the object's instance path of the configured type
- **In case the ElipseX is active**: When writing, this property works just like when the object is inactive. However, when reading this property returns the specified object, in case it exists. If the path does not point to an existing object at the time,

this property returns `Nothing`

CHAPTER 4

View

This section contains information about events, methods, and properties of E3's view objects:

- **Viewer**
- **Frames and Splitters**
- **Screen and Screen Objects**
- **E3Alarm**
- **E3Browser**
- **E3Chart**
- **E3Playback**
- **Reports**

4.1 Viewer

This section contains information about **events**, **methods**, and **properties** of the Viewer object.

4.1.1 Events

This section contains information about the events of the Viewer object.

4.1.1.1 OnInactive

OnInactive()

This event occurs while the Viewer is inactive, if the **EnableInactivity** property is set to True. It is triggered when it is checked that the user is not using the Viewer for a period of time equal or greater than the value of the **InactivityTime** property.

In a script for this event, the user may program what to do when the Viewer is inactive for a certain period of time. For example, it is possible to determine that after 20 minutes, a *logout* will be performed. Example:

```
Sub Viewer_OnInactive()
  Logout(false)
  If MsgBox("This Viewer section was closed due to inactivity.")
= 0 Then
    Application.GetFrame("").OpenScreen "InitialScreen", 0
End Sub
```

4.1.1.2 User Events

This section contains information about the user events of the Viewer object.

4.1.1.2.1 OnLogin

OnLogin()

Occurs when a user performs a sucessfully system *login* (user authentication). The system *login* can be performed using the **Login** method or when an object which can only be accessed by users with a certain authorization level requests authentication. Example:

```
Sub Viewer_OnLogin()
  MsgBox "Authorized user. Welcome to the system!"
End Sub
```

4.1.1.2.2 OnLogout

OnLogout()

Occurs when a *logout* is performed, that is, the user quits the system. The *logout* is performed by calling the **Logout** method. Example:

```
Sub Viewer_OnLogout()
  MsgBox "User quit the system."
End Sub
```

4.1.2 Methods

This section contains information about the methods of the Viewer object.

4.1.2.1 CaptureScreen

CaptureScreen(*Filename*)

The **CaptureScreen** method captures the current Screen and saves it in a file. This methods has the same effect of the PRINT SCREEN key in Windows. The *Filename* parameter determines the filename in which the Screen will be saved.

NOTE: The final format of the file is a Bitmap (.bmp), even if in the *Filename* parameter another file extension is informed (.gif, .jpg, etc.).

Example:

```
Sub CommandButton1_Click()
  Screen.Frame.CaptureScreen "c:\temp\screen.bmp"
End Sub
```

4.1.2.2 ChangePassword

ChangePassword()

This method opens a dialog box to change current user's password. This method returns True if the current user has permission to change their password. Otherwise, it returns False, indicating an operation failure, or that it is not possible to change the password, because the user is not authorized.

NOTE: This method is only available if the logged-in user belongs to an E3 Domain.

4.1.2.3 DoModal

DoModal(*Screen*, *Title*, *Left*, *Top*, *Width*, *Height*, *Arg*, *Flags*)

Opens a modal Screen, which is a window that do not allow clicking other Screens while it is not closed. The title parameter passed in this method will only be used if the **Caption** property is empty. Otherwise, the parameter will be ignored.

This method has the following parameters:

- ***Screen***: determines the name of the Screen
- ***Title***: determines the title of the modal window
- ***Left***, ***Top***: XY position of the modal window in pixels
- ***Width***: width of the modal window in pixels or HIMETRIC
- ***Height***: height of the modal window in pixels or HIMETRIC
- ***Arg***: determine variable to be used in the OnPreShow event of the Screen
- ***Flags***: determines a combination used on the modal window. Such combination is done by summing values of the next table that corresponds to the user options. When the specified value is equal to -1, the Viewer configurations are set to the modal window. When this value is different from -1, you may use the combinations described on the next table

Possible combinations for Flags parameter

VALUE	DESCRIPTION
1	Enables a title bar on the window.
2	Enables the **Close** button on the window.
4	Enables the **Minimize** button on the window.
8	Enables the **Maximize** button on the window.
16	Enables a window border.
32	Specifies that the window may be resized.

VALUE	DESCRIPTION
64	Specifies that the window may be moved.
256	Specifies that the window will stay on the top of the Screen.
512	Specifies that the window will be configured with a Tool Bar style.
1024	Disables object buttons.
2048	Centralizes the modal Screen in the Frame horizontally as well as vertically.

Example:

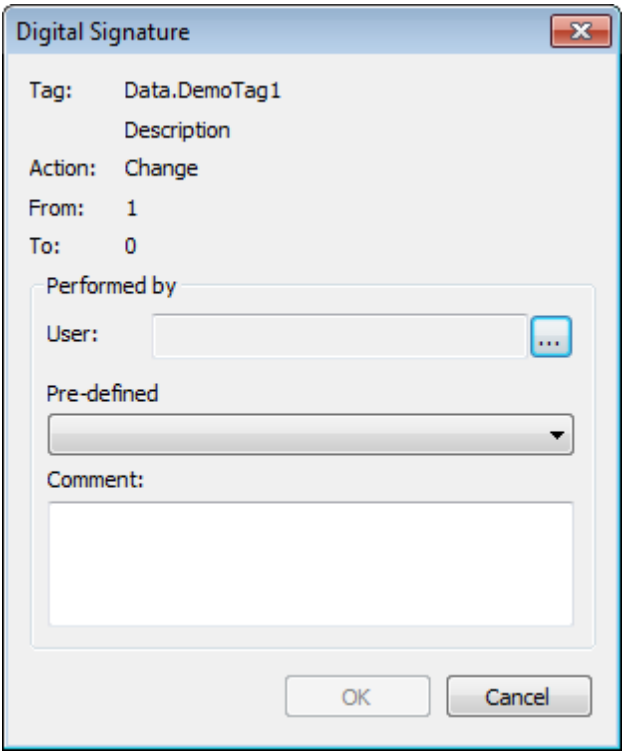
```
Sub Button1_Click()
  ' When clicking the Button1 opens another modal Screen
  Application.DoModal "Screen1", "Title", 0, 0, 400, 200, 0, 3
End Sub
```

NOTE: Size values on this method may be informed as numbers or **Strings**. In case of numbers, they are interpreted as pixels. In case of **Strings**, if followed by the "hm" unit, they are interpreted as Himetric. Any other case is considered as pixels.

4.1.2.4 ESign

ESign(*ObjName*[, *Description*[, *Action*[, *From*[, *To*[, *User*[, *Comment*]]]]]])

The **ESign** method is used to validate a field change, using an electronic signature. When this method is used, the following dialog box is displayed:



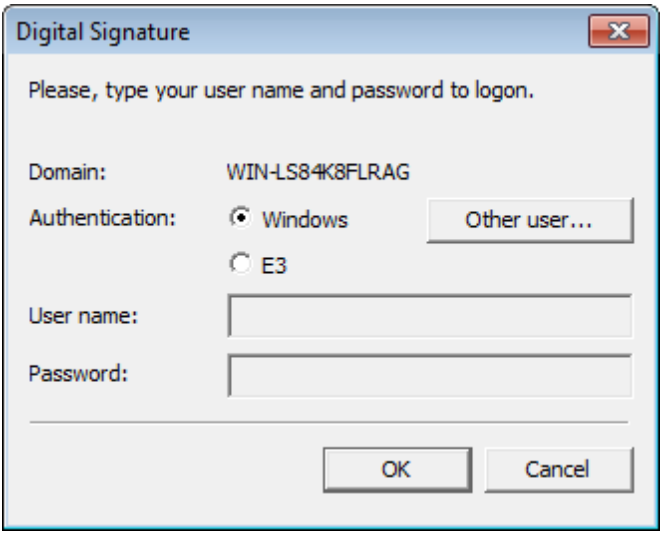
Digital Signature dialog box

Parameters of the ESign method

PARAMETER	DESCRIPTION
ObjName	Text that contains a Tag name or another application object.
Description	Text that contains the description of **ObjName**. This parameter is optional and if it is omitted, the dialog box will try to retrieve data from **ObjName**'s **DocString** property.
Action	Text that contains an action to be executed (for example, "Value change"). This parameter is optional and its default value is an empty **String**.
From	**Variant** which contains the original value, or the state to be altered. This parameter is optional.
To	**Variant** which contains the new Tag value, or the value to be applied in **Action**. This parameter is optional.

PARAMETER	DESCRIPTION
User	Return text. Receives the *login* name selected on the dialog box. This parameter is optional.
Comment	Return text. Receives a comment typed on the dialog box. This parameter is optional.

When clicking ..., a window then opens to authenticate the user. If option **Windows** is selected, fields **User name** and **Password** are automatically disabled. Click **Other user** to select a user belonging to the network domain. In case option **E3** is selected, type information about a user belonging to an E3 Domain in fields **User name** and **Password**.



Integrated login

This method returns True if the user clicks **OK**, and if the **User** and **Password** fields are valid. Otherwise, if the dialog box is dismissed or if the *login* or the password are wrong after three attempts, this method returns False. In case of failure, **User** and **Comment** are configured to an empty **String**.

Pre-defined comments are stored on Windows Registry. Only the last 26 comments are saved. Every time the window is created, the last comments are retrieved from Registry, and then used to fill in a list box. If users provide a new comment, it will be saved and the oldest one will be discarded, in case there is no free position. If it is an already existing comment, then it moves to the top of the recent comments list. Example:

```
Sub Button1_Click()
  Dim Tag, User, Comment
  Set Tag = Application.GetObject("IO.Inputs.IO01")
```

```
  If Application.ESign(Tag.PathName, , "Value Change", _
    Tag.Value, 1, User, Comment) Then
    If Tag.WriteEx = 1 Then
      Application.TrackEvent _
        "Tag IO.Inputs.IO01 changed to 1 " &_
        "by the user" & User, Comment
    End If
  End If
End Sub
```

4.1.2.5 ExecuteExternalApp

ExecuteExternalApp(*AppPath*, *Arguments*, *InitialDir*, *CmdShow*[, *ProcessId*])

This method executes an external application with *AppPath* name and path, and with *Arguments*, starting at the iniciando no diretório de trabalho *InitialDir* working directory. When a document is specified at *AppPath*, the application associated to this document is executed, and the document is passed as one of the application parameters. *ProcessID* receives a number that identifies the process (this number is used on **IsAppRunning** method and is the same value which appears on Windows Task Manager, on **PID** column). The *CmdShow* parameter specifies the application window opening mode, as described on the next table:

Available options for CmdShow parameter

OPTION	DESCRIPTION
0	Hides the window and activates another window.
1	Activates and displays the window. If the window is maximized or minimized, it will be restored to the original size and position. An application must specify this value when it is displaying a window for the first time.
2	Activates the window and displays it minimized.
3	Activates the window and displays it maximized.
4	Displays the window with its most recent size and position. The active window remains active.
5	Activates the window and display it with its current size and position.
6	Minimizes the window and activates the next higher level window.
7	Displays the window minimized. The active window remains active.
8	Displays the window in its current state. The active window remains active.

OPTION	DESCRIPTION
9	Activates and displays the window. If the window is maximized or minimized, it will be restored to its original size and position. An application must specify this value when restoring a window which was minimized.

Example:

```
Sub CommandButton1_Click()
  Dim ret
  Application.ExecuteExternalApp "calc.exe", "", "", 1, ret
  Application.GetObject("Data.InternalTag1").Value = ret
End Sub
```

NOTE: The parameter returned by *ProcessID* may be 0, in case any process has been started. For example, if an open document is an URL and an Internet Explorer instance is already running, it will display the document. No new process is started, thus *ProcessID* is 0.

4.1.2.6 Exit

Exit()

This method closes the window in Viewer. Example:

```
Sub_Button1.Click()
  Application.Exit()
EndSub
```

4.1.2.7 GetFormulaUnitDataObj

GetFormulaUnitDataObj(*FormulaName*)

This method gets the settings of the existing units on a certain Formula. The units are the destination of saved data in the Formula (values). This method has the *FormulaName* parameter, which informs Formula name.

Use the **GetFormulaUnitDataObj** method to get a collection of units of a Formula. This method returns True if the operation is successful, or False otherwise. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, displays a
  ' message box with the number of Units and the name
  ' of the First Unit
  Set obj = Application.GetFormulaUnitDataObj("Formula1")
  MsgBox CStr(obj.Count)
  MsgBox CStr(obj.Name(1))
End Sub
```

4.1.2.8 GetFormulaValueDataObj

GetFormulaValueDataObj(*FormulaName*)

This method gets the settings of the existing values of a certain Formula. Values are a data set saved in the Formula. This method has the *FormulaName* parameter, which informs Formula name.

Use the **GetFormulaValueDataObj** method to get a collection of values in the Formula. This method returns True if the operation is successful, or False otherwise. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, displays a message box
  ' with the number of Sets
  ' and the name of the First Set.
  Set obj = Application.GetFormulaValueDataObj("Formula1")
  MsgBox CStr(Obj.Count)
  MsgBox CStr(obj.Name(1))
End Sub
```

4.1.2.9 GetFrame

GetFrame([*FrameName*])

The **GetFrame** method searches for a splitter object which is already open in the current Viewer. This method has the *FrameName* parameter, which is optional and determines the frame name to be searched. In case the specified value at *FrameName* is empty, it will return a frame which contains all splitters or the active Screen. With the return value, use splitter methods, for example, the **OpenScreen** method to open another Screen. Example:

```
Sub Button1_Click()
  ' When clicking the button, gets the Menu frame
  ' and replaces the current Screen of this frame by the Options
screen
  Set frame = Application.GetFrame("Menu")
  ' frame has an object of type Splitter
  frame.OpenScreen "Options", 0
End Sub
```

4.1.2.10 GetFullUserName

GetFullUserName()

The **GetFullUserName** method returns the complete name of the user logged in E3. In case there is no logged-in user, an empty **String** is then returned.

4.1.2.11 GetKeyPad

GetKeyPad()

Returns a reference to an Elipse KeyPad object, it allows manipulating a floating virtual keyboard on applications developed using E3. See the **E3 User's Manual** for more information on this object. The methods and properties of the Elipse KeyPad are described on chapter **ActiveX - Elipse KeyPad**.

4.1.2.12 GetMouseX

GetMouseX()

Returns the X coordinate of the mouse, in pixels, relative to the whole area of the computer screen.

NOTE: This method fails in a script if the current mouse position cannot be retrieved. One of the situations where this failure can be verified is when the Windows Logon window is open (when pressing the CTRL + ALT + DEL key combination). This behavior of preventing access to the current mouse position is default on Windows and cannot be overridden. As a suggestion, use the `On Error Resume Next` command before using this method to prevent script errors.

4.1.2.13 GetMouseY

GetMouseY()

Returns the Y coordinate of the mouse, in pixels, relative to the whole area of the computer screen.

NOTE: This method fails in a script if the current mouse position cannot be retrieved. One of the situations where this failure can be verified is when the Windows Logon window is open (when pressing the CTRL + ALT + DEL key combination). This behavior of preventing access to the current mouse position is default on Windows and cannot be overridden. As a suggestion, use the `On Error Resume Next` command before using this method to prevent script errors.

4.1.2.14 GetValue

GetValue(*TagName*)

This method searches for an object value specified on *TagName* parameter. If *TagName* points to a property, this method returns the value of the property. However, if the *TagName* parameter specifies an object, this method returns the value of the **Value** property of the object. Example:

```
Sub Button1_Click()
  ' When clicking the button gets a tag value
  ' executed on a DataServer
```

```
  X = Application.GetValue("DataServer1.InternalTag1")
End Sub
```

4.1.2.15 IsAppRunning

IsAppRunning(*ProcessId*)

Indicates if an application started by the **ExecuteExternalApp** method is being executed. Returns True if the application identified on the operating system by *ProcessId* is running. Otherwise returns False. Example:

```
Sub CommandButton1_Click()
  Application.ExecuteExternalApp _
    "www.elipse.com.br", "", "", 1, processID
  While Application.IsAppRunning(processID)
    ' Waits for the application end
  Wend
  MsgBox "The application ended!"
End Sub
```

NOTE: The *ProcessId* parameter is the same value appearing in Windows Task Manager, in the **PID** column.

4.1.2.16 IsUserMemberOfGroup

IsUserMemberOfGroup(*GroupName*[, *UserName*])

This method checks if a user belongs to a certain group. It has the following parameters:

- **GroupName**: Name of the group to check
- **UserName**: Name of the user to check. If this parameter is omitted, or if it is an empty **String**, this method considers the user currently logged in the Viewer

This method returns True if the user belongs to the *GroupName* group, or False otherwise.

4.1.2.17 IsWebViewer

IsWebViewer()

Checks if the application is viewed by WebViewer. This method returns True if the application is being executed on WebViewer. Otherwise, returns False.

4.1.2.18 LoadFormulaDlg

LoadFormulaDlg(*FormulaName*[, *UnitName*[, *ValueName*]])

This method presents a dialog box which allows the user choosing a value set and

the destination unit, loading a Formula. This method has the *FormulaName* parameter, which determines the name of the formula object to be processed.

Use the **LoadFormulaDlg** method to call a dialog box for loading data of the Formula object specified by *FormulaName*. In this dialog box it is possible to specify which value set (*UnitName*) will be sent to which tag set (*ValueName*). In this message box the user has all value and unit sets available in the Formula object, and can freely set one to each other. When the user clicks **OK**, the value set is loaded on the specified unit. Example:

```
Sub Button1_Click()
  ' Calls the dialog box to operate
    Dim val
    Application.LoadFormulaDlg("Formula1")
End Sub
```

4.1.2.19 LoadFormulaValues

LoadFormulaValues(*FormulaName*, *UnitName*, *ValueName*)

This method automatically loads a value set to a destination unit, displaying a dialog box which allows the user informing different values from the ones defined on the Formula. This method has the following parameters: *FormulaName* determines the name of the formula and *UnitName* determines the name of the unit. The name of the value set is configured in the *ValueName* parameter.

NOTE: This method returns a logical value, that is, returns True if successful, and False on failure, which does not mean a script error.

Example:

```
Sub Button1_Click()
  Application.LoadFormulaValues "Formula1", "Unit1", "Value1"
End Sub
```

4.1.2.20 LoadFormulaValuesQuiet

LoadFormulaValuesQuiet(*FormulaName*, *UnitName*, *ValueName*)

Loads a value set to a destination unit, without any message. This method has he following parameters: *FormulaName* determines the name of the Formula, and *UnitName* which determines the unit name. The name of the value set is configured in the *ValueName* parameter. Example:

```
Sub Button1_Click()
  Application.LoadFormulaValuesQuiet "Formula1", "Unit3",
"Value1"
End Sub
```

NOTE: This method is also accessible through the **Formula** object.

4.1.2.21 LoadReport

LoadReport(*ReportName*)

Loads a Report template. The *ReportName* parameter is the name of the Report to be loaded. Example:

```
Sub Rect_Click()
  ' Loading a pre-defined report
  Set strRep = Application.LoadReport("[Report3]")
  strRep.PrintPreview ' Previewing print
End Sub
```

4.1.2.22 Login

Login([*Mode*])

Opens a dialog box for *login* (user authentication) in the application. The logged-in user remains in memory until another *login* or *logout* (when a user quits the application) are performed. This method has the *Mode* parameter, which is a Boolean determining whether a confirmation or failure message must be displayed (the default is False). When a Screen is about to be opened (using the **OpenScreen** method), a check on security settings is then performed. If True, the Screen is only opened if the logged-in user has permission. Otherwise, a *login* dialog box is opened.

4.1.2.23 LoginUser

LoginUser(*Username*, *UserPassword*)

Executes a *login* for a specific user, without displaying any message. The *Username* parameter is the name of the user, and the *UserPassword* parameter is the password of this user. This method returns True if the user *login* was successful, and False otherwise. If the user passed on *Username* parameter is configured to change the password on next *login*, this method returns False.

NOTE: This method is only available if the logged-in user belongs to an E3 Domain.

4.1.2.24 Logout

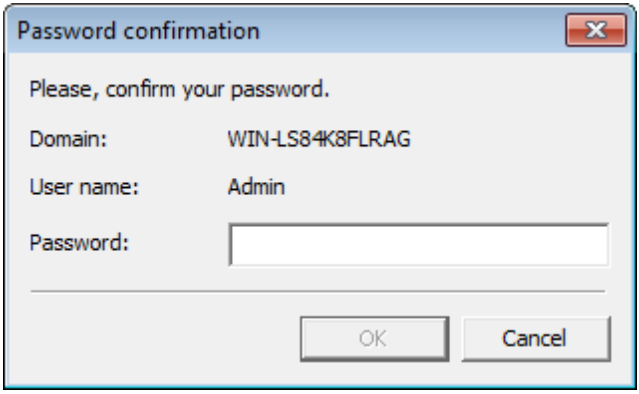
Logout([*Mode*])

Executes a *logout* (when the current user quits the application) from Viewer. In case there is no logged-in user, this method has no effect. From this moment on, it is considered that an "Anonymous" user is using the application. (Users can use the **OnLogout** event to perform a script to go to the initial Screen or end the application). This method has the optional *Mode* parameter, which is a **Boolean** determining whether a confirmation or failure message must be displayed (the default is False).

4.1.2.25 PasswordConfirm

PasswordConfirm(*Mode*)

This method opens a dialog box asking the currently logged-in user to retype the password. It returns True if the password is confirmed, or False otherwise. The **Boolean** *Mode* parameter determines whether a *logout* must be performed in case of failure.



Password confirmation

If the dialog box is closed by clicking **Cancel**, this method returns False. In case there is no logged-in user, this method returns False, but without opening a dialog box. In case the typed password is wrong, it is then asked three more times. If users type it incorrectly these three times, the dialog box is then closed and the method returns False.

NOTE: This method is only available if the logged-in user belongs to an E3 Domain.

4.1.2.26 Playsound

Playsound(*Filename*)

Plays a sound file which path and name are indicated in the *Filename* parameter. The file must follow these specifications:

- It must be Windows sound format (.wav extension)
- If the file is in the project (added using the **Insert resource**) command, the filename must be enclosed in brackets

If a folder was created on the project and the file was added using the **Insert resource** command, the path must be enclosed in quotation marks (for example, "c:\sound\ding.wav"). If the file is in the local directory, the name does not need to be enclosed in quotation marks, it is enough to type the path (for example, c:\sound

\ding.wav). Example:

```
Sub InitialScreen_OnAlarm()
  ' If there is an active alarm, an alert sound will be played.
  ' When the alarm is recognized, the sound stops.
  Set Alarm = Application._
    GetObject("AlarmConfig1.Area1.AlarmSource1")
    If Alarm.ActiveNACKAlarms = True Then
      Application.PlaySound("[ringin.wav]")
  End If
End Sub
```

4.1.2.27 SelectMenu

SelectMenu(*Menu*[, *Left*[, *Top*]])

This method displays a *pop-up* menu as specified by the *Menu* parameter. This parameter is a text with several options delimited by a vertical bar, each one of these **Strings** is a menu option. In case there is a set of two consecutive delimiters, a separator will be inserted. Use opening and closing braces to create a submenu. An asterisk in front of a **String** indicates an already marked option. An exclamation point disables the option.

The dialog box position can be set through the *Left* and *Top* parameters, which indicate the distance from Screen's left margin and top in pixels, respectively. In case these parameters are not informed, the menu is positioned according to the region where the mouse was clicked.

This method returns 0 if no option is selected, or the option number, 1 for the first option, 2 for the second, and so on. Example:

```
Sub Button1_Click()
  op = _
    Application.SelectMenu("Option1||Option2{*Option2|Option3}|
Option4|!Opption5")
  If op = 1 Then
    MsgBox "Option 1 was chosen"
  ElseIf op = 2 Then
    MsgBox "Option 2 was chosen"
  ElseIf op = 3 Then
    MsgBox "Option 3 was chosen"
  ElseIf op = 4 Then
    MsgBox "Option 4 was chosen"
  ElseIf op = 0 Then
    MsgBox "No option was chosen"
  End If
End Sub
```

4.1.2.28 SetValue

SetValue(*TagName*, *NewVal*)

This method sets the value of an object inside server. The **SetValue** method searches for an object or property executed on the server and sets the value specified in the *TagName* parameter. The type and value of the *NewVal* parameters must be supported by the object specified in *TagName*. Example:

```
Sub Button1_Click()
  ' When clicking Button1 sets value 20
  ' to Value property of the tag
  Application.SetValue "DataServer1.InternalTag1", 20
End Sub
```

4.1.2.29 ShowDatePicker

ShowDatePicker(*DateValue*, *Left*, *Top*[, *DefaultDate*])

Opens a dialog box to change date and time. This method returns True if the user confirms the date, or False if the user cancels editing. The new date is returned in the *DateValue* parameter. The position of the dialog box can be configured using the *Left* and *Top* parameters, which indicate the distance from left margin and top of the Screen in pixels, respectively. In case these parameters are not informed, the dialog box is centralized on the screen. The value of the *DefaultDate* parameter is the initial date and time when the dialog box was opened. If no date is informed, the system assumes current date. If no time is informed, it will start as 00:00:00. If no date nor time are informed, it will start with the current date and time. Example:

```
Sub Text2_Click()
  Dim newTime
  Application.ShowDatePicker newTime, 300, 300
  MsgBox "The time is : " & newTime
End Sub
```

4.1.2.30 ShowFilePicker

ShowFilePicker(*Open*, *FileName*[, *Extension*[, *Flags*[, *Filter*]]])

Displays the Windows Save or Open File dialog box. The *Open* parameter indicates the dialog box to be opened. If True, opens the Open File dialog box. If False, opens the Save dialog box. The *FileName* parameter indicates the variable where the file name to be saved or loaded will be stored, in case this method returns True. This parameter must be a variable. The *Extension* parameter is optional and informs the default file extension to be attached to the file name on the dialog box, when no extension is informed. In case it is empty, no extension will be attached at the end of the file name. Multiple extensions can be specified using a semicolon as the delimiter. The **String** must end with double vertical bars.

The *Flags* parameter is optional and defines the dialog box behavior. It is an integer, the sum of values from the next table. The *Filter* parameter is optional and defines a

set of **String** pairs specifying filters that can be applied to the files. The first **String** describes the filter, and the second one indicates the type of extension to use.

Possible combinations for Flags parameter

VALUE	DESCRIPTION
1	**CREATEPROMPT**: If the user specifies a non-existing file, this flag asks if the file must be created. If yes, the dialog box is closed and the file name is returned in the *Filename* parameter. Otherwise, the dialog box remains open.
2	**FILEMUSTEXIST**: Specifies that the user can only type existing file names. Otherwise, the dialog box displays a warning on the message dialog.
4	**NOCHANGEDIR**: Recovers the current directory to the original value in case the user have changed the directory while searching for the file. It has no effect for Open File in Windows XP.
8	**NODEREFERENCELINKS**: Sets the dialog box to return the selected shortcut file (.lnk). If this flag is not specified, the dialog box return the path and file name referenced by the *shortcut*.
16	**NOREADONLYRETURN**: Determines that the returned file is not read-only, and the directory is not record protected.
32	**PATHMUSTEXIST**: Specifies that the user can only inform valid files and directories, otherwise a message box is displayed to warn the user.
64	**READONLY**: Makes the option button **Read Only** to be initially selected when the dialog box is created.
128	**OVERWRITEPROMPT**: Makes the Save As dialog box generates a message informing that the file exists, and asks for confirmation to overwrite the file.

Filter example:

```
"Chart Files (*.xlc)|*.xlc|Excel spreadsheets (*.xls)|*.xls|Data
Files (*.xlc;*.xls)|*.xlc; *.xls|All files (*.*)|*.*||"
```

4.1.2.31 ShowPickColor

ShowPickColor(*ColorValue*, *Color*, *Left*, *Top*)

Opens the Windows Colors dialog box to choose a color. The decimal value of the chosen color is returned in the *ColorValue* parameter. The *Color* parameter indicates a previously selected color on the color palette. If this parameter is not informed, assumes black (0). The position of the dialog box can be set using the *Left* and *Top* parameters, which indicate the distance from the left margin and from the top of the screen, respectively, in pixels. In case these parameters are not informed, the dialog box is centralized on the screen. Example:

```
Sub CommandButton_Click()
  Dim newColor
  Dim defaultColor
  defaultColor = 65280 ' Light green
  If Application.ShowPickColor(newColor, defaultColor, 90, 90)
Then
    Screen.Item("Rectangle1").ForegroundColor = newColor
    Screen.Item("Text1").Value = newColor
  End If
End Sub
```

4.1.2.32 Stopsound

Stopsound()

Stops playing a sound. Example:

```
Sub CommandButton2_Click()
  Application.StopSound()
End Sub
```

4.1.2.33 ToggleValue

ToggleValue(*TagName*, *ValA*, *ValB*)

The **ToggleValue** method searches for an object value or property executed on the server and compare them to the *ValA* and *ValB* parameters. If the searched value is equal to *ValB*, the object or property specified in *TagName* will receive the value of *ValA*. Otherwise, it will receive the value of *ValB*. In case the value of *TagName* is not *ValA* nor *ValB*, the **ToggleValue** method will set the value specified in *ValA*. Example:

```
Sub Button1_Click()
  ' When clicking the button, sets the value
  ' to a tag executed on a DataServer.
  ' Sets a value of 20 to the tag.
  Application.SetValue "DataServer1.InternalTag1", 20
  ' As the value of InternalTag1 already is 20,
  ' the ToggleValue method alternates the value to 30.
  Application.ToggleValue "DataServer1.InternalTag1", 30, 20
```

```
End Sub
```

4.1.2.34 TrackEvent

TrackEvent(*EventMessage*, *Comment*, *TimeStamp*)

The **TrackEvent** method allows generating events manually via script. These events can be generated on the Viewer as well as on the Server, and are registered on an application's database table.

Parameters of the TrackEvent method

NAME	DESCRIPTION
EventMessage	Contains the event message (200 characters).
Comment	(Optional) Contains additional comments about the event (200 characters).
TimeStamp	(Optional) Indicates the date and time the event occurred. If it is not specified, E3 assumes current date and time.

The **TrackEvent** method will only record events in case the **Events Recording** option in the Domain Options is enabled. Events are recorded on a database table, which is also defined on the Event Recording settings.

For more information on Domain's Event Log, check the **E3 User's Manual**. Example:

```
Sub Button1_Click()
  Dim Tag, User, Comment
  Set Tag = Application.GetObject("IO.Inputs.IO01")
  If Application.ESign(Tag.PathName, , "Value Change", _
    Tag.Value, 1, User, Comment) Then
    If Tag.WriteEx 1 Then
      Application.TrackEvent _
        "Tag IO.Inputs.IO01 changed to 1 " &_
        "by the user" & User, Comment
    End If
  End If
End Sub
```

4.1.2.35 UserAdministration

UserAdministration()

This method opens a dialog box allowing to edit a list of Server's users. The available functions are:

- Display a list of all users
- Delete users (it is not possible to delete the current user)
- Add and edit users

- Edit user settings
- Change the user's password
- Change other user's data (*login*, name, etc.)

IMPORTANT: Only the Administrator has access to the **UserAdministration** method. The dialog box for user configuration is only accessible to a user enabled as Administrator.

4.1.3 Properties

This section contains information about the properties of the Viewer object.

4.1.3.1 BlinkTime

9 Defines the time, in milliseconds, between each state change when an object must blink (that is, each time an object has a link and the **Blink** option is checked). The default value of this property is 200 ms.

NOTE: The minimum time of screen refreshing is 55 ms; therefore, if this property is set with a time less than 55 ms, this configuration has no effect at all.

4.1.3.2 CacheEnable

◩ Keeps in memory the Screens already opened and instantiated on the Viewer, allowing a quick switching among them. If this property is enabled, then the Screen cache is also enabled.

4.1.3.3 Caption

A Determines the name of the application that is using Viewer. The default value of this property is an empty **String**.

4.1.3.4 CenterWindow

◩ When enabled, determines that the Viewer's window must start centered. Otherwise, the default settings will be used. The default value of this property is True.

4.1.3.5 CloseButton

◩ If this option is enabled, the **Close** button is enabled on the Viewer, and it can be used. Otherwise, this button does not display on the window. The default value of this property is True.

4.1.3.6 CommErrorBkColor

This property is used to define the background color of a Setpoint when a Link or Connection fails. The default value of this property is red (**RGB(255, 0, 0)**). Also check the **EnableCommError** property.

4.1.3.7 CommErrorText

This property is used to define the alert message when a Link or Connection fails. The default value of this property is "???". Also check the **EnableCommError** property.

4.1.3.8 CommErrorTextColor

This property is used to define the text color of a Setpoint when a Link or Connection fails. The default value of this property is yellow (**RGB(255, 255, 0)**). Also check the **EnableCommError** property.

4.1.3.9 DisableTaskSwitching

If it is set to True, disables window switching on the Viewer. Otherwise, window switching is enabled. The default value of this property is False. This property can be changed at run time, by calling the **SetDisableTaskSwitching** method.

4.1.3.10 EnableCommError

Enables or disables viewing communication errors. For example, if there is a Setpoint on the Screen which is linked to an I/O Tag, and the communication between E3 and this Tag fails, the text configured in the **CommErrorText** property is displayed in the SetPoint, with the color configured in the **CommErrorTextColor** property, and the background color of the SetPoint defined on the **CommErrorBkColor** property. The default value of this property is True.

4.1.3.11 EnableHeartbeat

Enables or disables sending a *heartbeat* (message sent on fixed intervals, indicating that Server is active) between Viewer and server. In case the Viewer stops receiving *heartbeat* messages, this means that some problem occurred, therefore connection must be aborted. The default value of this property is False.

4.1.3.12 EnableInactivity

Enables or disables checking the user inactivity period. For more information, see the Viewer's **OnInactive** event. The default value of this property is False.

4.1.3.13 EnableZoomMenu

◩ If it is set to True, enables displaying a screen zoom configuration menu using the right mouse button at run time, except when a script is configured with a different information on the **MouseDown** or **MouseUp** events. Otherwise, the menu will not be displayed. The default value of this property is True.

4.1.3.14 HeartbeatPeriodMs

9 Indicates the interval (in milliseconds) between *heartbeat* messages sent by the Server. The *heartbeat* message is always sent when the server remains a period indicated by this property without sending messages for the Viewer. The default value of this property is 2000 (two seconds).

4.1.3.15 HeartbeatTimeoutMs

9 Indicates the time, in milliseconds, that the Viewer waits without receiving messages from an Server. If this time passes, and no message is received, the Viewer assumes that the connection was lost, and starts the reconnection process. This time must be greater than the time set in the **HeartbeatPeriodMs** property, preferably greater than the double. The default value of this property is 5000 (five seconds).

4.1.3.16 InactivityTime

9 Defines the maximum waiting time for a mouse or keyboard event before the inactivity period, in minutes. For more information, see the Viewer's **OnInactive** event. The default value of this property is 5 (five) minutes. Example:

```
Sub CommandButton3_Click()
  MsgBox "The application will remain inactive for " & _
    Application.InactivityTime & " minute(s)."
End Sub
```

4.1.3.17 InitialScreen

A Indicates the initial Screen or Frame displayed when the Viewer is called. Using the **WindowStyle** property, it is possible to determine if the window must start maximized, windowed, or minimized. The default value of this property is "InitialScreen".

4.1.3.18 IsPlaybackMode

◩ If True, indicates that the Viewer is running inside an E3Playback object, in *playback* mode. This is a read-only property and it is only available at run time.

4.1.3.19 IsReadOnly

◩ If set to True, indicates that the Viewer is in **Read-Only** mode (restricted access).

4.1.3.20 LoginRetries

9 Specifies the number of Viewer's *login* retries, that is, how many times the *login* dialog box will be displayed besides the first time. The default value of this property is 2 (two).

4.1.3.21 MaximizeButton

◩ If this option is enabled (True), the **Maximize** button is enabled on the Viewer, and this button can be used. Otherwise, this button does not display on the window. The default value of this property is True.

4.1.3.22 MinimizeButton

◩ If this options is enabled (True), the **Minimize** button is enabled on the Viewer, and this button can be used. Otherwise, this button does not display on the window. The default value of this property is True.

4.1.3.23 Params

A This property is a vector of key-value pairs, which returns parameters passed to the Viewer through the command line option **-params**. All values are returned as **Strings**. For example, if the Viewer command line contains the following parameters:

```
Viewer -params Language=ENU
```

Users can use the following code to check what is the Viewer's start up language.

```
Sub InitialScreen_OnStartRunning()
  Select Case Application.Params("Language")
    Case "ENU"
      Item("Text1").Value = "English"
    Case Else
      Item("Text1").Value = "Unrecognized language"
  End Select
End Sub
```

NOTE: The key is not case sensitive (users can use "Language" or "language"), but returned values, specially if used with a **Select** command, are case sensitive.

4.1.3.24 ReconnectDialogDelaySec

9 Indicates the number of seconds the Viewer will wait during a possible reconnection to the server, before displaying a message warning the user about this action (this property does not affect the first connection). If it is equal to 0, the reconnection message is always displayed. To avoid displaying this message, it is advisable to put a very large number (one billion, for example).

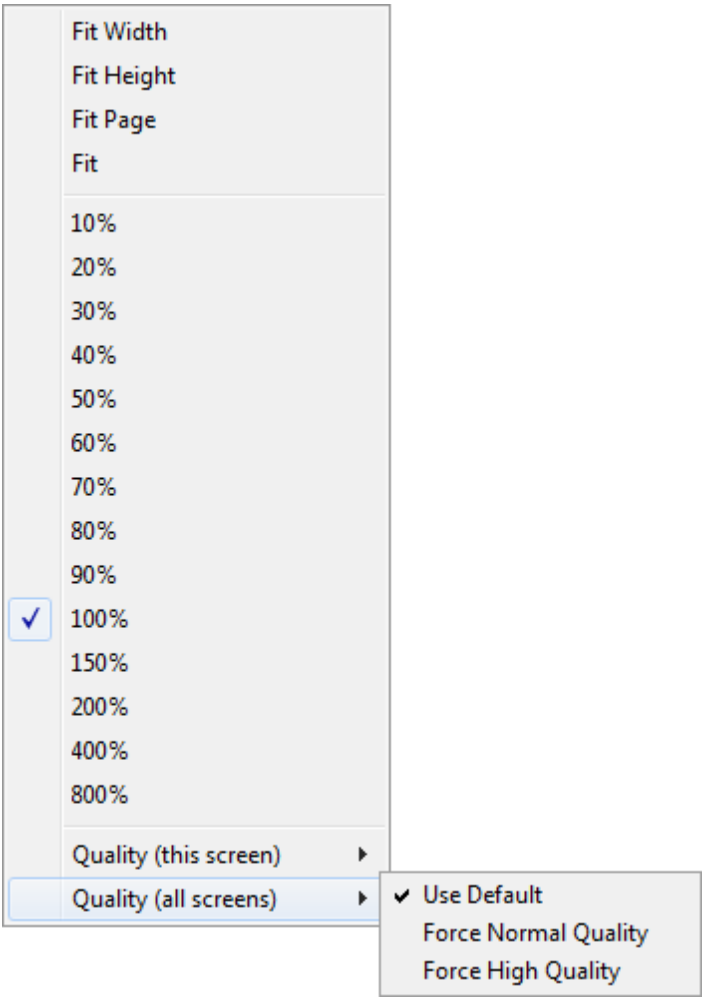
NOTE: When reconnection occurs silently, all windows from the active Viewer are disabled and an hourglass is displayed indicating that the application is unavailable. During reconnection time, users are not allowed to cancel the process.

4.1.3.25 RenderQuality

Controls the drawing quality of all Screens, only if the value of **RenderQuality** property of the Screens is equal to **0 - rqDefault**. Possible values for this property are the following:

- **0 - rqDefault**: Uses a normal quality mode, known as GDI (*Graphics Device Interface*). Corresponds to item **Use Default** from Viewer's contextual menu **Quality (all screens)**, at run time. This is the default value for applications created on versions prior to 4.0
- **1 - rqNormal**: Forces a normal quality mode (GDI) for drawing all Screens. Corresponds to item **Force Normal Quality** from Viewer's contextual menu **Quality (all screens)**, at run time
- **2 - rqHighQuality**: Forces a high quality mode (GDI+) for drawing all Screens. Corresponds to item **Force High Quality** from Viewer's contextual menu **Quality (all screens)**, at run time. This is the default value for applications starting at version 4.0

The following figure displays the Viewer's contextual menu at run time, with the respective configuration options for this property:



Viewer's contextual menu at run time

4.1.3.26 ShowKeyPadOnEdit

This property automatically enables displaying the Elipse KeyPad when a Screen object, which allows editing, gets the focus.

4.1.3.27 TargetDPIX

Defines the dots per inch value, horizontally, of the destination computer's monitor. The default value of this property is -1, which means that the current computer's value is assumed.

4.1.3.28 TargetDPIY

9 Defines the dots per inch value, vertically, of the destination computer's monitor. The default value of this property is -1, which means that the current computer's value is assumed.

4.1.3.29 TargetMarginX

9 Defines the number of pixels that must be discounted of the Screen's horizontal resolution (the Viewer's working area). The default value of this property is -1, which assumes the Viewer's window configuration (with or without a title bar, with or without border) together with the current computer's configuration (border width and title bar defined by Windows).

4.1.3.30 TargetMarginY

9 Defines the number of pixels that must be discounted of the Screen's vertical resolution (the Viewer's working area). The default value of this property is -1, which assumes the Viewer's window configuration (with or without a title bar, with or without border) together with the current computer's configuration (border width and title bar defined by Windows)

4.1.3.31 TargetResolutionX

9 Defines the Screen's horizontal resolution that this application is designed to (in pixels). The default value of this property is -1, which assumes the current computer's resolution.

4.1.3.32 TargetResolutionY

9 Defines the Screen's vertical resolution that this application is designed to (in pixels). The default value of this property is -1, which assumes the current computer's resolution.

4.1.3.33 TitleBar

◩ If this option is enabled (True), the Viewer's title bar is displayed, according to the specifications of the **Caption** property. Otherwise, it is hidden. The default value of this property is True.

4.1.3.34 User

A Contains the name of the user currently using the Viewer. This is a read-only property.

4.1.3.35 ViewerLanguageId

9 Returns the running Viewer's language code. The available values of this property are described on the next table. This is a read-only property and it is only available at run time.

Available options for ViewerLanguageId

DECIMAL	HEXADECIMAL	LANGUAGE
4	0x0004	Chinese Simplified
1031	0x0407	German
1033	0x0409	English (US)
1034	0x040A	Spanish
1046	0x0416	Brazilian Portuguese

NOTE: This property **DOES NOT** correspond to the Windows installation language, neither the language configured in the Windows **Regional and Language Options** (**Clock, Language and Region** in Windows 7) control panel.

4.1.3.36 WindowBorder

◩ Enables or disables a border around the Viewer's window. The default value of this property is True. This is a read-write property, but changing its value at run time does not change the settings of an already opened Viewer, only for windows opened using the Viewer's settings.

4.1.3.37 WindowHeight

A Determines the height of the Viewer's window, in pixels. The default value of this property is 300.

4.1.3.38 WindowMovable

◩ Indicates whether the window can be moved. The default value of this property is True. This is a read-write property, but changing its value at run time does not change the settings of an already opened Viewer, only for windows opened using the Viewer's settings.

4.1.3.39 WindowResizable

◩ Indicates whether the window can be resized. This property is effective only if the **WindowBorder** property is True. The default value of this property is True. This is a read-write property, but changing its value at run time does not change the settings of an already opened Viewer, only for windows opened using the Viewer's settings.

4.1.3.40 WindowSmallTitle

Indicates whether the Viewer's window should have a small title bar. This property is effective only if the **TitleBar** property is True. The default value of this property is False. This is a read-write property, but changing its value at run time does not change the settings of an already opened Viewer, only for windows opened using the Viewer's settings.

4.1.3.41 WindowStayOnTop

Indicates whether the Viewer's window must always be on top of other windows. The default value of this property is False. This is a read-write property, but changing its value at run time does not change the settings of an already opened Viewer, only for windows opened using the Viewer's settings.

4.1.3.42 WindowStyle

Defines the initial style of the Viewer's window. The available options are:

- **0 - Maximized**: starts Viewer maximized
- **1 - Windowed**: starts Viewer windowed
- **2 - Minimized**: starts Viewer minimized

4.1.3.43 WindowWidth

Determines the width of the Viewer's window, in pixels. The default value of this property is 400.

4.2 Frames and Splitters

This section contains information about **methods** and **properties** of the Splitter object and about **properties** of the Frame object. The Splitter object does not have events associated to it and the Frame object does not have events nor methods associated to it.

4.2.1 Splitter Methods

This section contains information about the methods of the Splitter object.

4.2.1.1 BringToFront

BringToFront()

Brings to the front a Splitter that is hidden or under another one. Example:

```
Sub Button1_Click()
  Application.GetFrame("Test").BringToFront()
```

```
End Sub
```

4.2.1.2 CaptureScreen

CaptureScreen(*Filename*)

Captures the content of a Splitter, saving it on the file pointed by *Filename*, in BMP format. Example:

```
Sub CommandButton1_Click()
  ' When clicking the button, copies the content
  ' of the splitter to Frame.bmp file.
  Screen.Frame.CaptureScreen ("c:\temp\frame.bmp")
End Sub
```

4.2.1.3 Close

Close(*Code*)

Use the **Close** method to close the Frame window. The *Code* parameter has the return value for the **DoModal** method, if this window has been called by this method. Example:

```
Sub CloseButton_Click()
  ' When clicking the button, closes the window.
  Screen.Close(0)
End Sub
```

4.2.1.4 FlashWindow

FlashWindow(*Number*, *Time*)

This method makes the Viewer icon on the Windows Task Bar start flashing. The *Number* parameter determines the number of times the Task Bar should flash and *Time* determines the time (in milliseconds) between flashes. Example:

```
Sub Text1_Click()
  Set frame = Application.GetFrame("_top")
  frame.FlashWindow 50, 500
End Sub
```

4.2.1.5 MaximizeFrame

MaximizeFrame()

Maximizes a Frame or modal Screen. Example:

```
Sub CommandButton4_Click()
  Application.GetFrame("Other").MaximizeFrame()
End Sub
```

4.2.1.6 MinimizeFrame

MinimizeFrame()

Minimizes a Frame or modal Screen. Example:

```
Sub CommandButton4_Click()
  Application.GetFrame("Other").MinimizeFrame()
End Sub
```

4.2.1.7 MoveFrame

MoveFrame([*PosX*[, *PosY*[, *SizeX*[, *SizeY*]]]])

Moves and resizes a Frame to a specific coordinate and size. The *PosX* and *PosY* parameters inform the new position, in pixels, relative to the left and top, respectively. The *SizeX* and *SizeY* parameters inform new size and height,respectively, in pixels or Himetric. All parameters are optional. Example:

```
Sub screen2_OnPreShow(vArg)
  ' When the Screen is opened on Test frame, changes the
position
  ' and the frame size
  Application.GetFrame("Test").MoveFrame 100, 100, 200, 200
End Sub
```

NOTE: The size values of this method can be informed as numbers or **Strings**. In case of numbers, they are considered as pixels. In case of **Strings**, if they are followed by the "hm" unit, they will be interpreted as Himetric. Any other case is considered as pixels.

4.2.1.8 OpenScreen

OpenScreen(*ScreenName*, *Arg*)

The **OpenScreen** method opens a Screen inside the Splitter. The *ScreenName* parameter determines the name of the Screen to be opened. It is also possible to specify a zoom percentage of the Screen, and also enable a scroll bar, by using the "?" key such as in the following template:

```
<screen-name>?<zoom>?<enable-scroll>
```

Where `screen-name` is the name of the Screen to be opened; `zoom` is the zoom percentage and `enable-scroll` enables or disables scrolling.

The zoom percentage of the Screen can have the following values:

- **1**: the whole page
- **2**: the Screen width occupies 100% of the Splitter's width, with proportional height
- **3**: the Screen height occupies 100% of the Splitter's height, with proportional

width

- **4**: The Screen completely fills the Splitter
- **5 to 100**: it is the equivalent of the zoom percentage of the Screen itself

Enabling scroll can have the following values:

- **0**: disables scrolling
- **1**: enables scrolling

The *Arg* parameter allows passing the specified value to the Screen, by using the **OnPreShow** event. Example:

```
Sub Button1_Click()
  ' When clicking the button opens Screen2 on Test frame
  ' and passes 1 to be used in the OnPreShow event
  Application.GetFrame("Test")._
    OpenScreen "Screen2?100?0", "This is a test."
End Sub

Sub Screen2_OnPreShow(vArg)
  ' The message box will display the sentence
  ' "This is a test."
  MsgBox vArg
End Sub
```

4.2.1.9 Refresh

Refresh(*Force*)

The **Refresh** method allows forcing a redrawn of a Screen's or Splitter's content. It should be used in Viewer scripts with massive processing (loops, for example), or in method calls demanding a long time and also demanding a visual indication of the progress of the process to the user.

Due to the general redrawn being a heavy operation, the default version of the **Refresh** method (without parameters) is optimized to ignore redrawn requests from E3. This standard behavior is ideal for loop progress indications, where a lot of redrawing is performed. The *Force* parameter disables this optimization, making sure that for each call to **Refresh** method, a redrawn is performed. However, when using this option, the **Refresh** method cannot be called repeatedly, such as inside a loop. Example:

```
Sub CommandButton1_Click()
  ' Draws a progress bar of an operation
    Screen.Item("Rectangle2")_
      .HorizontalPercentFill = (i / 30) * 100
    Frame.Refresh True
    ' <-- some lengthy operation -->
```

```
    i = i + 1
  Wend
End Sub
```

4.2.1.10 RestoreFrame

RestoreFrame()

Allows restoring the Frame window to its original size. Example:

```
Sub CommandButton1_Click()
  Application.GetFrame("Other").RestoreFrame()
End Sub
```

4.2.1.11 SetDisableTaskSwitching

SetDisableTaskSwitching(*Disable*)

Enables or disables window switching. The *Disable* parameter is a **Boolean** value that indicates whether the window switching is enabled or not. This method updates the Viewer's **DisableTaskSwitching** property.

NOTES:

- If more than one Viewer instance is running on the same machine, and at least one of these instances use the **SetDisableTaskSwitching** method, that change affects all Viewers on this machine
- In case it is necessary to change the window title or style, the **SetFrameOptions** method must be used after calling the **SetDisableTaskSwitching** method

4.2.1.12 SetForegroundWnd

SetForegroundWnd()

The **SetForegroundWnd** method activates and sets the focus of the Viewer window. This method is useful for grabbing the operator's attention about an event, when the Viewer window is hidden or minimized. Example:

```
Sub CommandButton1_Click()
  Application.GetFrame("Other").SetForegroundWnd()
End Sub
```

4.2.1.13 SetFrameOptions

SetFrameOptions(*Title*, *Flags*)

Used to configure the Frame title on the window and the window style. The *Title* parameter is a **String** that contains the window title. This text is displayed if the Screen's **Caption** property is empty.

The *Flags* parameters specifies the window style. If this parameter is omitted, the default value is -1. This value is used to keep the previous configuration of the window. When this value is not -1, users can change the window style by specifying

a sum of values of the following combinations:

Possible combinations for the Flags parameter

VALUE	DESCRIPTION
1	Enables the window title bar.
2	Enables the **Close** button on the window.
4	Enables the **Minimize** button on the window.
8	Enables the **Maximize** button on the window.
16	Enables the window border.
32	Specifies that the window can be resized. To do so, the window must have a border.
64	Specifies that the window can be moved.
256	Specifies that the window will stay on top of the Screen.
512	Specifies that the window will be configured as a Tool Bar style.
1024	Disables object buttons.
2048	Centers the window.

Example:

```
Sub Screen_OnPreShow()
  Frame.SetFrameOptions("Alarm Screen", 114)
End Sub
```

In the previous example, the 114 (2 + 16 + 32 + 64) value indicates that the window has a **Close** button enabled (2), a border (16), can be resized (32), and can be moved (64). The window title is "Alarm Screen".

In the **Open Screen** and **Open Modal Screen** Picks, it is also possible to configure the window style during edition, by using the Window Style dialog box. For more information, see the topic **Picks**.

NOTE: The **SetFrameOptions** method must be used after calling the **SetDisableTaskSwitching** method, in case there is a need to change the window title or style.

4.2.2 Splitter Properties

This section contains information about the properties of the Splitter object.

4.2.2.1 IsHTML

The **IsHTML** property returns True if the Splitter contains HTML code inserted into the Frame. Otherwise, returns False.

4.2.2.2 SplitBorder

Enables or disables the Splitter border, and determines whether the border between main and remaining Splitters should be displayed at run time. This property has no effect on the remaining Splitter. The default value of this property is True.

4.2.2.3 SplitDockPosition

Indicates the Splitter position on the Screen. The available options are:

Available options for SplitDockPosition

OPTION	DESCRIPTION
0 - dockRemaining	Places the Splitter as the remaining one, that is, it occupies the remaining space in the horizontal or vertical Splitter.
1 - dockTop	Places the Splitter as the main one, above the remaining.
2 - dockBottom	Places the Splitter as the main one, below the remaining.
3 - dockLeft	Places the Splitter as the main one, on the left of the remaining.
4 - dockRight	Places the Splitter as the main one, on the right of the remaining.

4.2.2.4 SplitLink

The **SplitLink** property contains a *link* that can be displayed in the Splitter. It is possible to specify a project Screen, an executable, or an Internet *link*. In case of Screens, it is possible to specify a zoom percentage and also enabling scroll bars using a "?" key, such as in the template `<screen-name>?<zoom>?<scroll-bar>`, where `screen-name` is the name of the Screen to be opened; `zoom` is the zoom percentage, and `scroll-bar` can be **1** to enable or **0** to disable. The `zoom` and `scroll-bar` properties are valid only if the indicated *link* is a Screen. Otherwise, they will be ignored. If the `zoom` parameter is not informed, assumes 100%. If the `scroll-bar` parameter is not informed, assumes **1**, that is, enabled. Example:

```
Sub CommandButton1_Click()
  Application.GetFrame("Splitter1").SplitLink = "Screen1?10?1"
End Sub
```

4.2.2.5 SplitResizable

◩ Determines whether the main Splitter can be resized at run time. This property has no effect on the remaining Splitter. The default value of this property is True.

4.2.2.6 SplitValue

9 The **SplitValue** property determines the value to be set to the Frame Splitter, which might be **%** (percentage), **hm** (HIMETRIC), or **px** (pixels). If the unit is omitted, it will be considered as Himetric. Exemplo:

```
Sub Splitter1_Click()
  SplitValue = 10
End Sub
```

4.2.3 Frame Properties

This section contains information about the properties of the Frame object.

4.2.3.1 Caption

A The **Caption** property defines the Frame title to be displayed on the Viewer's title bar.

4.3 Screens and Screen Objects

This section contains information about events, methods, and properties of Screens and Screen Objects.

4.3.1 Screen

This section contains information about **events**, **methods**, and **properties** of the Screen object.

4.3.1.1 Events

This section contains information about the events of the Screen object.

4.3.1.1.1 Click

Click()

It occurs when the left mouse button is pressed on a Screen. This event will not occur if the screen is not visible, or if its **Enabled** property is set to False. The Screen visibility depends on three factors: **Visible** property set to True; parent object visible, and object's **Layer** property on the Screen layer. Example:

```
Sub Screen_Click()
  ' Displays a message box when
```

```
  ' the user clicks the screen
  MsgBox "You clicked the screen."
End Sub
```

4.3.1.1.2 DbClick

DbClick()

Occurs when there is a left mouse button double-click on a Screen. This event will not occur if the object is not visible, or if its **Enabled** property is set to False. The object's visibility depends on three factors: **Visible** property set to True; parent object visible, and object's **Layer** property on the Screen layer. Example:

```
Sub Screen_DbClick()
  ' Displays a message box when
  ' the user double-clicks the screen
  MsgBox "You double-clicked the screen."
End Sub
```

4.3.1.1.3 KeyDown

KeyDown(*KeyCode*, *Shift*)

It occurs when a key is pressed, regardless of the Screen focus.

Variables of the KeyDown event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

Example:

```
Sub Screen1_KeyDown(KeyCode, Shift)
  ' Displays a message box when
  ' the user presses a key
  MsgBox "Key code: " & KeyCode
End Sub
```

4.3.1.1.4 KeyUp

KeyUp(*KeyCode*, *Shift*)

It occurs when a key is released, regardless of the Screen focus.

Variables of the KeyUp event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

Example:

```
Sub Screen1_KeyUp(KeyCode, Shift)
  ' Displays a message box when the user
  ' releases a key
  MsgBox "Key code: " & KeyCode
End Sub
```

4.3.1.1.5 MouseDown

MouseDown(*Button*, *ShiftState*, *MouseX*, *MouseY*)

It occurs when any mouse button is pressed on a Screen. Use the MouseDown event to determine specific actions when the user clicks a Screen.

Variables of the MouseDown event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the pressed key along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays the Screen's X coordinate where the mouse was clicked.
MouseY	Displays the Screen's Y coordinate where the mouse was clicked.

Example:

```
Sub InitialScreen_MouseDown(Button, ShiftState, MouseX, MouseY)
  ' Quits the application when a mouse click occurs
  ' on the InitialScreen object.
  Application.Exit()
End Sub
```

4.3.1.1.6 MouseUp

MouseUp(*Button*, *ShiftState*, *MouseX*, *MouseY*)

It occurs when a previously clicked mouse button is released. Use the **MouseUp** event to determine specific actions to be triggered only when a mouse button is released.

Variables of the MouseUp event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the pressed key along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays the Screen's X coordinate where the mouse was clicked.
MouseY	Displays the Screen's Y coordinate where the mouse was clicked.

Example:

```
Sub InitialScreen_MouseUp(Button, ShiftState, MouseX, MouseY)
  ' Quits the application only when the user releases the mouse
button.
  Application.Exit()
End Sub
```

4.3.1.1.7 OnHide

OnHide()

It occurs when the Screen is about to be closed. Use the **OnHide** event to perform any operations before closing the Screen object. This event may occur on several situations:

- When the Screen is replaced by another one, by using the **OpenScreen** method
- When the user closes the window where the Screen is
- When the **Close** method is called from the Screen object
- When the Viewer is closed or quit

Example:

```
Sub InitialScreen_OnHide()
```

```
  Application.Exit()
End Sub
```

4.3.1.1.8 OnPreShow

OnPreShow(Arg)

It occurs before displaying the Screen. The *Arg* variable receives the content of the *Arg* parameter of the **OpenScreen** method, which generates this event. After that, the **OnShow** event is generated. Example:

```
Sub Screen1_OnPreShow(Arg)
  ' The title of Screen1 to be displayed
  ' was passed as a parameter when calling the OpenScreen
  ' method that generated the event.
  Caption = Arg
End Sub
```

4.3.1.1.9 OnShow

OnShow()

It occurs in the very moment when a Screen is displayed. Use the **OnPreShow** event to perform any operation before displaying this Screen. Example:

```
Sub MainScreen_OnShow()
  MsgBox "Welcome to the system!"
End Sub
```

4.3.1.2 Methods

This section contains information about the methods of the Screen object.

4.3.1.2.1 Close

Close(*Code*)

Use the **Close** method to close the Screen. This method generates the **OnHide** event before being performed. The *Code* parameter will contain the return value for the **DoModal** method, if the Screen is called by this method. Example:

```
Sub CloseButton_Click()
  ' When CloseButton is clicked, closes the window
  Screen.Close(0)
End Sub
```

4.3.1.2.2 FromPixelX

FromPixelX(*XPixel*)

Converts the Screen's X coordinate, indicated by the *XPixel* parameter, from pixels to Himetric. This method is complementary to the **ToPixelX** method.

4.3.1.2.3 FromPixelY

FromPixelY(*YPixel*)

Converts the Screen's Y coordinate, indicated by the *YPixel* parameter, from pixels to Himetric. This method is complementary to the **ToPixelY** method.

4.3.1.2.4 ToPixelX

ToPixelX(*XHimetric*)

Converts the Screen's X coordinate, indicated by the *XHimetric* parameter, from Himetric to pixels. This method is complementary to the **FromPixelX** method.

4.3.1.2.5 ToPixelY

ToPixelY(*YHimetric*)

Converts the Screen's Y coordinate, indicated by the *YHimetric* parameter, from Himetric to pixels. This method is complementary to the **FromPixelY** method.

4.3.1.3 Properties

This section contains information about the properties of the Screen object.

NOTE: E3 uses the HIMETRIC system to define coordinates and widths. In this system, each logical unit is the equivalent of a thousandth of a centimeter, that is, every 1,000 units correspond to one centimeter. Thus, this is the standard adopted in the description of E3 properties, when applicable.

4.3.1.3.1 BackgroundColor

Specifies the Screen's background filling color. In scripts, use the VBScript's **RGB** method to gather the color to be associated to this property. The default value of this property is gray (**RGB(192, 192, 192)**).

4.3.1.3.2 Caption

The **Caption** property defines the Screen title to be displayed on the Viewer's title bar.

4.3.1.3.3 FillStyle

This property specifies the filling style of the Screen and Screen objects. The next table contains valid values for this property.

Available options for FillStyle

OPTION	DESCRIPTION
0 - Solid	Filling is solid (default).
1 - Hollow	No filling. This value is not available for the Screen object.
2 - Horizontal	Filling has horizontal stripes.
3 - Vertical	Filling has vertical stripes.
4 - Downward	Filling has down stripes from left to right in a 45 degrees angle.
5 - Upward	Filling has up stripes from left to right in a 45 degrees angle.
6 - Cross	Filling has horizontal and vertical stripes.
7 - DiagonalCross	Filling has diagonal stripes from left to right in a 45 degrees angle.
8 - Gradient	Filling has a gradient using **ForegroundColor** as well as **BackgroundColor**. The effect will be defined by the **GradientStyle**property.
9 - SemiTransparent	Leaves the object translucent. This value is not available for the Screen object.
10 - MouseArea	Filling is empty, but the object still responds to events (default). This value is not available for the Screen object.
11 - Background	Fills the Screen with background color.
12 - Picture	Fills the Screen with the picture selected on **PictureFile** property. This value is only available for the Screen object.

NOTE: The **FillStyle** property is not available for the **Picture** object.

4.3.1.3.4 ForegroundColor

Specifies the Screen's foreground color. In scripts, use the VBScript's **RGB** method to gather the color to be associated to this property. The default value of this property is black (**RGB(0, 0, 0)**). Applications previous to the introduction of this property will have both **ForegroundColor** and **BackgroundColor** properties set to the color stored on the **BackgroundColor** property, and the filling style set to **11 - Background**, which paints the whole Screen with the background color (old behavior, before creating styles). Example:

```
Sub Screen1_Click()
  ' Changes the foreground color to blue
  ForegroundColor = RGB(0, 0, 255)
End Sub
```

4.3.1.3.5 GradientStyle

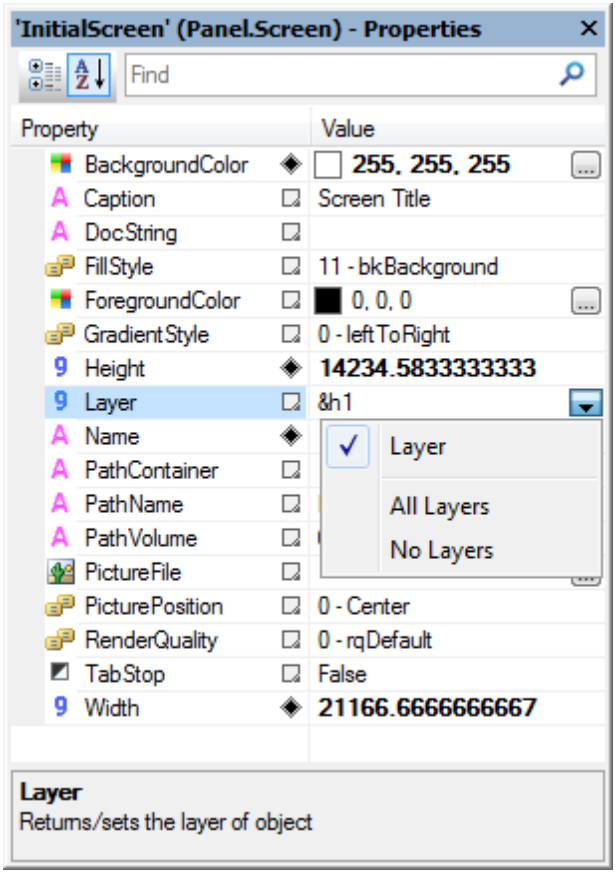
This property specifies the Screen's gradient filling style. This property will only be used when the **FillStyle** property is set to 8 (**Gradient**). The gradients consider the color change as starting at **ForegroundColor** and ending at **BackgroundColor**.

Available options for GradientStyle

OPTION	DESCRIPTION
0 - LeftToRight	Vertical gradient, from left to right.
1 - RightToLeft	Vertical gradient, from right to left.
2 - VerFromCenter	Vertical gradient from center to border.
3 - VerToCenter	Vertical gradient from border to center.
4 - BottomUp	Horizontal gradient from bottom to top.
5 - TopDown	Horizontal gradient from top to bottom.
6 - HorzFromCenter	Horizontal gradient from center to border.
7 - HorzToCenter	Horizontal gradient from border to center.
8 - DiagUpRight	Diagonal gradient to the top with foreground color on the right (default).
9 - DiagUpLeft	Diagonal gradient to the top with foreground color on the left.
10 - DiagUpFromCenter	Diagonal gradient to the top from center to border.
11 - DiagUpToCenter	Diagonal gradient to the top from border to center.
12 - DiagDownLeft	Diagonal gradient to the bottom with foreground color on the left.
13 - DiagDownRight	Diagonal gradient to the bottom with foreground color on the right.
14 - DiagDownFromCenter	Diagonal gradient to the bottom from center to border.
15 - DiagDownToCenter	Diagonal gradient to the bottom from border to center.
16 - SpotSouthEast	Gradient with foreground color from right lower corner.
17 - SpotSouthWest	Gradient with foreground color from left lower corner.
18 - SpotNorthWest	Gradient with foreground color from left upper corner.
19 - SpotNorthEast	Gradient with foreground color from right upper corner.
20 - SpotFromCenter	Gradient with background color from center to border.
21 - SpotToCenter	Gradient with background color from border to center.

4.3.1.3.6 Layer

9 This property defines in which layers the object should appear. Its value represents a 32-bit mask, one bit for each layer. Therefore, up to 32 layers can be defined. Thus, objects may be logically grouped and displayed or hidden only by modifying the **Layer** property mask.



Layer options

Example:

```
Sub Screen1_Click()
  Layer = 1
End Sub
```

NOTE: The object's visibility depends on three factors: the **Visible** property must be set to True; the parent object must be visible, and the object's **Layer** property must be enabled for the Screen.

4.3.1.3.7 PictureFile

Contains the picture's filename that will be used as the Screen's background. It can be any format supported by E3 on DrawPicture objects (*.bmp, *.gif, *.jpg, *.cur, *.ico, *.emf, *.wmf). The default value of this property is an empty **String**. This property is only valid if the **FillStyle** property is set to **12 - Picture**.

4.3.1.3.8 PicturePosition

Indicates the position of the selected picture on the Screen's **PictureFile** property. This property is only valid if the **FillStyle** property is set to **12 - Picture**. The valid options are:

Available options for PicturePosition

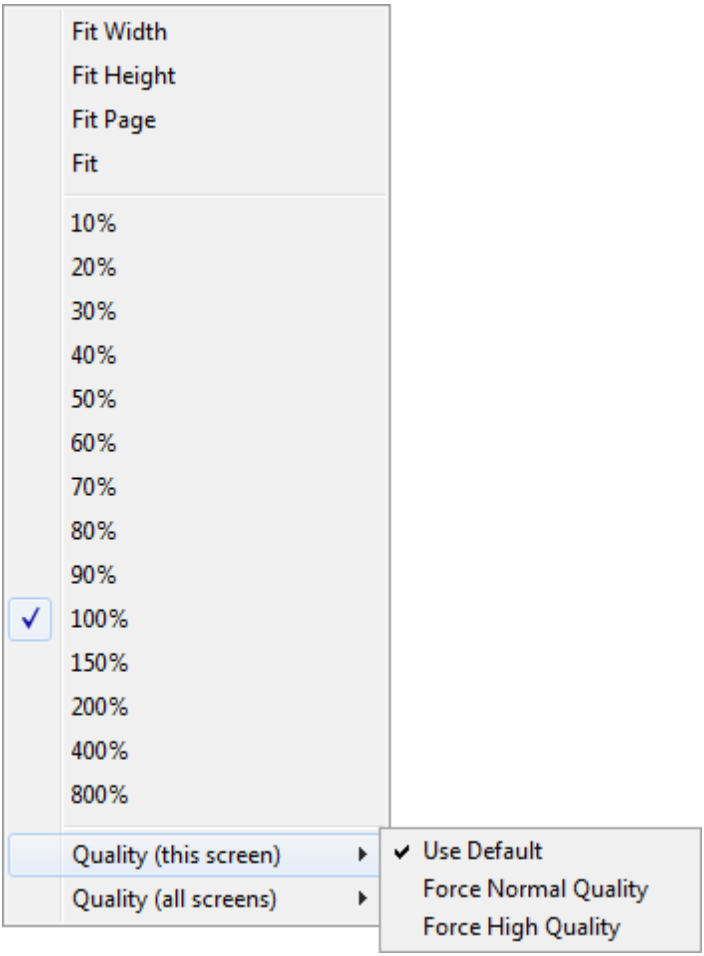
OPTION	DESCRIPTION
0 - Center	Picture with original size, centered on Screen.
1 - Tile	Picture with original size, repeated as many times as needed to fill the Screen.
2 - Stretch	Picture resized to fill the Screen.
3 - TopLeft	Picture with original size, on the left upper corner of the Screen.
4 - BottomLeft	Picture with original size, on the left lower corner of the Screen.
5 - BottomRight	Picture with original size, on the right lower corner of the Screen.
6 - TopRight	Picture with original size, on the right upper corner of the Screen.

4.3.1.3.9 RenderQuality

Controls the drawing quality of a Screen. Possible values for this property are the following:

- **0 - rqDefault**: This Screen's drawing quality uses the value defined on Viewer's **RenderQuality** property. This is the default values of this property, even in application created on versions prior to 4.0
- **1 - rqNormal**: Forces a normal quality mode (GDI) when drawing this Screen
- **2 - rqHighQuality**: Forces a high quality mode (GDI+) when drawing this Screen

The following figure displays a Screen's contextual menu at run time, with the respective configuration options for this property:



Screen's contextual menu at run time

4.3.2 Screen Objects

This section contains information about events, methods, and properties of Screen Objects.

4.3.2.1 Common Events

This section contains information about common events of all Screen Objects.

4.3.2.1.1 Click

Click()

It occurs when the left mouse button is pressed on an object. This event will not

occur if the object is not visible, or if its **Enabled** property is set to False. The visibility of an object depends on three factors: **Visible** property set to True; parent object visible, and the object's **Layer** property present on the Screen layer. Example:

```
Sub Button_Click()
  ' Displays a message box when
  ' the user clicks an object
  MsgBox "You clicked an object."
End Sub
```

4.3.2.1.2 DbClick

DbClick()

It occurs when there is a double click, that is, the left mouse button is quickly pressed twice on an object. This event will not occur if the object is not visible, or if its **Enabled** property is set to False. The visibility of the object depends on three factors: **Visible** property set to True; parent object visible, and object's **Layer** property present on Screen layer. Example:

```
Sub Button_DbClick()
  ' Displays a message box when
  ' the user performs a double-click over the object
  MsgBox "You clicked twice over the object."
End Sub
```

4.3.2.1.3 KeyDown

KeyDown(*KeyCode*, *Shift*)

It occurs when a key is pressed and the object has the keyboard focus. Notice that this event will not be generated if the object is not enabled (its **Enabled** property set to False) or if this object does not have keyboard focus.

Variables of the KeyDown event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

Example:

```
Sub Button_KeyDown(KeyCode, Shift)
  ' Displays a message box when
  ' the user presses a key
  MsgBox "Key code: " & KeyCode
End Sub
```

4.3.2.1.4 KeyUp

KeyUp(*KeyCode*, *Shift*)

It occurs when a key is released and the object has keyboard focus. Notice that this event will not be generated if the object is not enabled (its **Enabled** property set to False) or if this object does not have keyboard focus.

Variables of the KeyUp event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

Example:

```
Sub Button_KeyUp(KeyCode, Shift)
  ' Displays a message box when the user
  ' releases a key
  MsgBox "Key code: " & KeyCode
End Sub
```

4.3.2.1.5 MouseDown

MouseDown(*Button*, *ShiftState*, *MouseX*, *MouseY*)

It occurs when any mouse button is pressed on an object.

Variables of the MouseDown event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the key pressed along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays the Screen's X coordinate where the mouse was clicked.
MouseY	Displays the Screen's Y coordinate where the mouse was clicked.

Example:

```
' Quits the application when there is a
' mouse click on the InitialScreen object.
Sub InitialScreen_MouseDown(Button, ShiftState, MouseX, MouseY)
  Application.Exit()
End Sub
```

4.3.2.1.6 MouseUp

MouseUp(*Button*, *ShiftState*, *MouseX*, *MouseY*)

It occurs when any previously clicked mouse button is released on an object. Use the **MouseUp** event to specify actions to be performed only when the mouse button is released.

Variables of the MouseUp event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the pressed key along with the mouse: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays the Screen's X coordinate where the mouse was clicked.
MouseY	Displays the Screen's Y coordinate where the mouse was clicked.

Example:

```
' Quits the application only when the user releases the mouse
button
' after clicking Rectangle1 object.
Sub Rectangle1_MouseUp(Button, ShiftState, MouseX, MouseY)
  Application.Exit()
End Sub
```

4.3.2.2 Common Methods

This section contains information about common methods of all Screen Objects.

4.3.2.2.1 BringToFront

BringToFront()

Brings the object to the uppermost layer on the Screen, in front of all other objects. Example:

```
Sub Button1_Click()
  ' When clicking Button1, the system
  ' brings Rectangle1 to the front
  Screen.Item("Rectangle1").BringToFront()
End Sub
```

4.3.2.2.2 SendToBack

SendToBack()

Sends the object to the lowermost layer on the Screen, at the back of all other objects. Example:

```
Sub Button2_Click()
  ' When clicking Button2, the system
  ' sends Rectangle1 to the back
  Screen.Item("Rectangle1").SendToBack()
End Sub
```

4.3.2.2.3 SetFocus

SetFocus()

Use the **SetFocus** method to move mouse or keyboard focus to a specific object. Example:

```
Sub Screen1_OnShow()
  ' When opening the window, moves focus to Button1
  Item("Button1").SetFocus()
End Sub
```

4.3.2.3 Common Properties

This section contains all properties common to all Screen objects. The properties of this section do not apply to the following objects: **ActiveX (MSForms)**, **E3Chart**, **E3Browser**, and **E3Alarm**. These objects are treated later, on specific chapters.

NOTE 1: E3 uses the Himetric system to define coordinates and widths. In this system, each logical unit is the equivalent of a thousandth of a centimeter, that is, every 1,000 units correspond to one centimeter. Thus, this is the standard adopted in the description of E3 properties, when applicable.

NOTE 2: The following properties are common to all previously cited objects, including Groups of objects and Sliders.

4.3.2.3.1 Angle

The **Angle** property defines the rotation angle in degrees, counter clockwise, that the object should be rotated. This property also applies to children objects of this one, respecting rotation limitations of each child object. The object rotates according to its center, which can be edited during the rotation operation. The default value of this property is 0 (no rotation).

4.3.2.3.2 BackgroundColor

This property specifies the object's background color. This color is used when the **BackgroundStyle** property is configured to **1** (opaque) and one of the **VerticalPercentFill** or **HorizontalPercentFill** properties has a value different from 100. Another use of this color is when the **FillStyle** property is configured with values between 2 and 8. This makes the remaining area use the background color for filling. In scripts, use the VBScript's **RGB** function to create the color associated to this property. The default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.3.3 BackgroundStyle

This property specifies the object's background filling mode. This property enables using the **VerticalPercentFill** and **HorizontalPercentFill** properties with values different from 100, and also set the **FillStyle** property with values between 2 and 8. This makes the remaining area use the background color configured in the **BackgroundColor** property for filling. The following table contains valid values for the **BackgroundStyle** property.

Available options for BackgroundStyle

OPTION	DESCRIPTION
0 - Transparent	No background is drawn.
1 - Opaque	If visible, the background is drawn.

4.3.2.3.4 BorderColor

Specifies the object's border or line color. This property is only used when the **BorderStyle** property is not configured to **5** (null), when the object does not have a border. In scripts, use the VBScript's **RGB** function to create the color to be associated to this property. The default value of this property is white (**RGB(255, 255, 255)**), except for Display and SetPoint objects, which default value is dark gray (**RGB(128,128,128)**).

4.3.2.3.5 BorderStyle

The **BorderStyle** property determines the object's border style.

Available options for BorderStyle

OPTION	DESCRIPTION
0 - Normal	Applies a solid border to the object (default).
1 - Dash	Applies a dashed border to the object.
2 - Dot	Applies a dotted border to the object.
3 - Dashdot	Applies a dash-dot border to the object.
4 - Dashdotdot	Applies a dash-dot-dot border to the object.
5 - Null	The object has no border.

4.3.2.3.6 BorderWidth

Defines the width, in Himetric units, of the object's border or line. It is only used if the **BorderStyle** property is not configured to **5** (null). The default value of this property is 0 (zero). This is an exception in E3's measuring system, because when its value is 0, the line or border width is not defined in Himetric units, but in pixels. Using the **BorderWidth** property with a value equal to zero indicates a one pixel width.

4.3.2.3.7 Effect3D

Applies a 3D effect in the selected object. The available options for this property are described in the following table.

Available options for Effect3D

OPTION	DESCRIPTION
0 - No3D	Transparent (default).
1 - Raised	Applies a raised 3D effect.
2 - Sunken	Applies a sunken 3D effect.

4.3.2.3.8 Effect3D_X

Specifies the 3D effect dimension in the object's horizontal axis (the X axis). The default value of this property is 30.

4.3.2.3.9 Effect3D_Y

Specifies a 3D effect dimension in the object's vertical axis (the Y axis). The default value of this property is 30.

4.3.2.3.10 Effect3DColorBase

Determines the color of the object's 3D base effect. The default value of this property is black (**RGB(0, 0, 0)**).

4.3.2.3.11 Effect3DColorTop

Determines the color of the object's 3D top effect. The default value of this property is white (**RGB(255, 255, 255)**).

4.3.2.3.12 Enabled

Enables or disables access via keyboard or mouse. This property only affects objects which allows input via keyboard or mouse.

4.3.2.3.13 FillStyle

This property specifies the object's filling style. The following table contains valid values for the **FillStyle** property.

Available options for FillStyle

OPTION	DESCRIPTION
0 - Solid	Filling is solid (default).
1 - Hollow	No filling.
2 - Horizontal	Filling has horizontal stripes.
3 - Vertical	Filling has vertical stripes.
4 - Downward	Filling has stripes down from left to right in 45 degrees.
5 - Upward	Filling has stripes up from left to right in 45 degrees.
6 - Cross	Filling has horizontal and vertical stripes.
7 - DiagonalCross	Filling has stripes up and down from left to right in 45 degrees.
8 - Gradient	Filling has a gradient using both the **ForegroundColor** and the **BackgroundColor** property. The effect is determined by the **GradientStyle** property.
9 - SemiTransparent	The object remains semi transparent.
10 - MouseArea	No filling, but the object remains responsive to events.

4.3.2.3.14 ForegroundColor

This property specifies the object's foreground color. This color is used when the **FillStyle** property is set to 0 (solid) or between 2 and 9. In scripts, use the VBScript's

RGB function to create the color to be associated to this property. The default value of this property is blue (**RGB(0, 0, 255)**), except for Display and SetPoint objects, which default value is white (**RGB(255, 255, 255)**).

4.3.2.3.15 Frame

Returns the object's parent Frame. This property is only accessible at run time.

4.3.2.3.16 GradientStyle

This property specifies the object's gradient filling style. This property is only used when the value of the **FillStyle** property is set to **8** (**Gradient**). The gradients consider the change as starting from the color configured in the **ForegroundColor** property and moving towards the color configured in the **BackgroundColor** property.

Available options for GradientStyle

OPTION	DESCRIPTION
0 - LeftToRight	Vertical gradient from left to right.
1 - RightToLeft	Vertical gradient from right to left.
2 - VerFromCenter	Vertical gradient from center to border.
3 - VerToCenter	Vertical gradient from border to center.
4 - BottomUp	Horizontal gradient from bottom to top.
5 - TopDown	Horizontal gradient from top to bottom.
6 - HorzFromCenter	Gradient from center to border.
7 - HorzToCenter	Gradient from border to center.
8 - DiagUpRight	Diagonal gradient to top with foreground color on the right (default).
9 - DiagUpLeft	Diagonal gradient to top with foreground color on the left.
10 - DiagUpFromCenter	Diagonal gradient to top from center to border.
11 - DiagUpToCenter	Diagonal gradient to top from border to center.
12 - DiagDownLeft	Diagonal gradient to bottom with foreground color on the left.
13 - DiagDownRight	Diagonal gradient to bottom with foreground color on the right.
14 - DiagDownFromCenter	Diagonal gradient to bottom from center to border.
15 - DiagDownToCenter	Diagonal gradient to bottom from border to center.
16 - SpotSouthEast	Gradient with foreground color from lower right corner.
17 - SpotSouthWest	Gradient with foreground color from lower left corner.
18 - SpotNorthWest	Gradient with foreground color from upper left corner.

OPTION	DESCRIPTION
19 - SpotNorthEast	Gradient with foreground color from upper right corner.
20 - SpotFromCenter	Gradient with background color from center to border.
21 - SpotToCenter	Gradient with background color from border to center.

IMPORTANT: A large amount of objects simultaneously displayed with a gradient may lead to a poor performance when updating the Screen. Using figures may solve this problem.

4.3.2.3.17 HasFocus

This property determines that the selected object has the focus. This property is only accessible at run time.

4.3.2.3.18 Height

Determines the object's height.

4.3.2.3.19 HorizontalFillStyle

Defines the object's horizontal filling. This property works along with the **HorizontalPercentFill** property, which informs the percentage of the object to be filled. These two properties allows a simulation of a level filling in an object, such as in a tank, for example.

Available options for HorizontalFillStyle

OPTION	DESCRIPTION
0 - FillLeftToRight	The filling percentage is from left to right (default).
1 - FillRightToLeft	The filling percentage is from right to left.
2 - FillCenterToEdgesH	The filling percentage is from center to border.

4.3.2.3.20 HorizontalPercentFill

Use the **HorizontalPercentFill** property to specify what is the percentage of the object's horizontal area to be filled. Acceptable values for this property vary from 0 to 100. This property works along with the **HorizontalFillStyle** property, which informs how the filling is going to be performed. The default value of this property is 100.

4.3.2.3.21 Layer

9 This property defines in which layers the object should appear. The value represents a 32-bit mask, one bit for each layer. Therefore, up to 32 individual layers can de defined. Thus, objects can be logically grouped and displayed or hidden only by changing the **Layer** mask property.

4.3.2.3.22 MouseOver

◩ Informs whether mouse pointer is on Screen. If True, **MouseOver** is enabled. Otherwise, it is not. This property is only available at run time. Default value is False.

4.3.2.3.23 MouseOverChild

◩ Informs whether mouse pointer is on one of the objects inserted on the Screen. If True, **MouseOverChild** is enabled. Otherwise, it is not. This property is only available at run time. Default value is False.

4.3.2.3.24 Screen

Returns the object's parent Screen. This property is only accessible at run time.

4.3.2.3.25 Shadow

◩ Indicates a shadow effect on the object. If True, the object has a shadow, whose coordinates are established by the **ShadowX** and **ShadowY** properties. Otherwise, the object does not have a shadow. The default value of this property is False.

4.3.2.3.26 ShadowColor

Specifies the object's shadow filling color. This color is used when the **Shadow** property is set to True. In scripts, use the VBScript's **RGB** function to create the color to be associated to this property. The default value of this property is dark gray (**RGB(128, 128, 128)**).

4.3.2.3.27 ShadowX

9 Defines the left vertical coordinate of the object shadow, in Himetric units. This shadow is always relative to the object's **X** property. Positive values indicate a shadow on the right side of the object, and negative ones on the left side. The default value of this property is 200.

4.3.2.3.28 ShadowY

Defines the upper horizontal coordinate of the object shadow, in Himetric units. This shadow is always relative to the object's **Y** property. Positive values indicate a shadow below the object, and negative ones above. The default value of this property is 200.

4.3.2.3.29 TabStop

This property determines how TAB key is used on the application. If it is True, the key can be used, otherwise it cannot be used.

4.3.2.3.30 Tip

The **Tip** property displays a popup text when mouse is on an object, at run time. Example:

```
Sub RoundRect1_MouseUp(nButton, nShiftState, x, y)
  Tip = "This is a test!"
End Sub
```

4.3.2.3.31 VerticalFillStyle

Defines the object's vertical filling. This property works along with the **VerticalPercentFill** property, which informs the percentage of the object to be filled. These two properties allow a simulation of a level filling in an object.

Available options for VerticalFillStyle

OPTION	DESCRIPTION
0 - FillBottomToTop	The filling percentage is from bottom to top.
1 - FillTopToBottom	The filling percentage is from top to bottom.
2 - FillCenterToEdgesV	The filling percentage is from center to border.

4.3.2.3.32 VerticalPercentFill

Use the **VerticalPercentFill** property to specify what percentage of the object's vertical area to fill. Acceptable values for this property vary from 0 to 100. This property works along with the **VerticalFillStyle** property, which indicates how this filling occurs. The default value of this property is 100.

4.3.2.3.33 Visible

Defines the object's visibility. If True, the object is visible, as long as the following factors are also present: this object's parent object must be visible, and this

object's **Layer** property must also be available on the Screen layer.

4.3.2.3.34 Width

Determines the object's width, in Himetric units.

4.3.2.3.35 X

The **X** property defines the object's left horizontal coordinate, in Himetric units.

4.3.2.3.36 Y

The **Y** property defines the object's upper vertical coordinate, in Himetric units.

4.3.2.4 Group

This section contains information about **properties** of the Group of objects. This object does not have events nor methods associated to it.

4.3.2.4.1 Properties

This section contains information about the properties of the Group of objects.

4.3.2.4.1.1 EnableOverrideLineColor

This property enables or disables the Group object to override the original colors of the lines of the objects contained in the Group. If the **EnableOverrideLineColor** is enabled, the original colors of the lines of the objects in the Group are replaced by the color set in the **OverrideLineColor** property. Otherwise, each object in the Group displays its original color. The default value of this property is False.

4.3.2.4.1.2 OverrideFillColor

When the **OverrideFillMode** property is set to 2 or 3, the **OverrideFillColor** property is used to define the filling color of the adjacent objects in the Group, instead of their original colors. In scripts, use the VBScript's **RGB** function to create the color to be associated to this property. The default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.4.1.3 OverrideFillMode

The **OverrideFillMode** property specifies the filling mode of the objects in the Group. It changes the original filling mode of the image, without changing the object's original filling settings.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	Original filling of the object.
1 - WireFrame	The objects are not filled, they only draw their borders.
2 - SolidFill	The filling of the objects in the Group is solid with the color specified in the **OverrideFillColor** property.
3 - ByBrightness	The filling of the objects in the Group is solid with the color specified in the **OverrideFillColor** property, however, the brightness of the original color of each object is preserved.

4.3.2.4.1.4 OverrideLineColor

When the **EnableOverrideLineColor** property is set to True, the **OverrideLineColor** property is used to define the color of the line of the objects in the Group, instead of their original color. In scripts, use the VBScript's **RGB** function to create the color to be associated to this property. The default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.5 Round Rectangle

This section contains information about the **properties** of the Rounded Rectangle object. This object does not have events nor methods associated to it.

4.3.2.5.1 Properties

This section contains information about the properties of the Round Rectangle object.

4.3.2.5.1.1 RoundAspectX

The **RoundAspectX** property defines the dimension of rectangle sizes on X axis. Thus, according to the value defined on this property, the corners of the rectangle will change their shape, from a rectangle to an ellipse. This property may vary its value from 0,1 to 1. Example:

```
Sub RoundRect_Click()
  RoundAspectX = 0.5
End Sub
```

4.3.2.5.1.2 RoundAspectY

The **RoundAspectY** property defines the size of the height dimension of the rectangle on Y axis. Thus, according to the value defined on this property, the

rectangle corners will change their shape, from a rectangle to an ellipse. This property may vary its value from 0,1 to 1. Example:

```
Sub RoundRect_Click()
  RoundAspectY = 0.5
End Sub
```

4.3.2.6 Arc of Ellipse

This section contains information about the **properties** of the Arc of Ellipse object. This object does not have events nor methods associated to it.

4.3.2.6.1 Properties

This section contains information about the properties of the Arc of Ellipse object.

4.3.2.6.1.1 ArcBeginAngle

This property configures the arc's initial angle, in degrees. The accepted interval of this property ranges from 0 to 359. The arc style and shape also depend on the **ArcEndAngle** and **ArcStyle** properties. The default value of this property is 0. Example:

```
Sub CommandButton9_Click()
  Screen.Item("Arc1").ArcBeginAngle = 12
End Sub
```

4.3.2.6.1.2 ArcEndAngle

This property configures the arc's final angle, in degrees. The accepted interval of this property ranges from 0 to 359. The arc style and shape also depend on the **ArcBeginAngle** and **ArcStyle** properties. The default value of this property is 270. Example:

```
Sub CommandButton9_Click()
  Screen.Item("Arc1").ArcEndAngle = 12
End Sub
```

4.3.2.6.1.3 ArcStyle

This property specifies the arc's border or line style. The object border will be drawn according to the defined style, using the color specified in **BorderColor** with the **BorderWidth** width. The next table contains valid values for the **ArcStyle** property:

Available options for ArcStyle

OPTION	DESCRIPTION
0 - arc	Drawing style is an arc.
1 - chord	Drawing style is a chord, connecting a starting and an ending point.

OPTION	DESCRIPTION
2 - pie	Drawing style is a pie (default).

Example:

```
Sub CommandButton9_Click()
  Screen.Item("Arc1").ArcStyle = 1
End Sub
```

4.3.2.7 Picture

This section contains information about the **properties** of the Picture object. This object does not have events nor methods associated to it.

4.3.2.7.1 Properties

This section contains information about the properties of the Picture object.

4.3.2.7.1.1 BackgroundColor

This property specifies the object's background filling color. This color is used when the **BackgroundStyle** property is set to 1 (opaque) and one of the **VerticalPercentFill** or **HorizontalPercentFill** properties is different from 100. Another use of this color is when the **FillStyle** property is set to a value between 2 and 8. This makes the remaining area uses the background color as the filling color. In scripts, use the VBScript's **RGB** method to build the associated color of this property. The default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.7.1.2 BackgroundStyle

This property specifies the object's background filling mode. This property enables using the **VerticalPercentFill** and **HorizontalPercentFill** properties with values different from 100 and also sets the **FillStyle** property with values between 2 and 8. This makes the remaining area to use the **BackgroundColor** property as the filling. The next table contains valid values for **BackgroundStyle** property.

Available options for BackgroundStyle

OPTION	DESCRIPTION
0 - Transparent	No object background will be drawn.
1 - Opaque	In case it is visible, the background will be drawn.

4.3.2.7.1.3 Convert

This property allows converting a picture. If its value is set to 0, it is possible to preview the conversion. Otherwise, it is not possible to preview the conversion. This

property only accepts 0 and 1. The default value of this property is 0.

4.3.2.7.1.4 EnableOverrideLineColor

This property enables or disables the object to overwrite the original color of the image line by the color defined in the **OverrideLineColor** property. If the **EnableOverrideLineColor** property is enabled, makes the original color of the object line to be modified by the color of the **OverrideLineColor** property. Otherwise, the Picture object displays the original color.

4.3.2.7.1.5 Filename

Defines the name of the image file associated to this object. The file path can be the complete disk file path, as well as the path relative to the application (when the image file is inserted as an application resource). The default value of this property is empty. The following image file types are supported:

Available image types for the Filename property

PROPERTY	FILTER	DESCRIPTION	FILTER
Bitmap file	BMP	No	Yes
Graphics Interchange Format	GIF	No	Yes
Joint Picture Expert Group	JPG	No	Yes
Icon File	ICO	No	Yes

4.3.2.7.1.6 HorizontalFillStyle

Defines how the object's horizontal filling will be performed. This property works with the **HorizontalPercentFill**, property, which informs what object percentage should be filled. These two properties allow simulating a level filling in an object, the same way as a tank level, for example.

Available options for HorizontalFillStyle

OPTION	DESCRIPTION
0 - FillLeftToRight	The filling percentage will be from left to right (default).
1 - FillRightToLeft	The filling percentage will be from right to left.
2 - FillCenterToEdgesH	The filling percentage will be from center to border.

Example:

```
Sub Circle1_OnStartRunning()
  HorizontalFillStyle = 2
```

```
End Sub
```

4.3.2.7.1.7 HorizontalPercentFill

Use the **HorizontalPercentFill** property to specify the percentage of the object's horizontal area to be filled. Acceptable values for this property range from 0 to 100. This property works with the **HorizontalFillStyle** property, which informs how this filling will be performed. The default value of this property is 100. Example:

```
Sub Circle1_OnStartRunning()
  HorizontalPercentFill = 200
End Sub
```

4.3.2.7.1.8 OverrideFillColor

When the **OverrideFillMode** property is set to 2 or 3, the **OverrideFillColor** property will be used to set the color of the image filling, instead of the original color. In scripts, use the VBScript's **RGB** method to build a color to be linked to this property. The default value of this property is red (**RGB(255, 0, 0)**). Example:

```
Sub DrawPicture1_Click()
  ' When clicking the object sets the Override
  ' mode to solid and changes the image filling
  ' color to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.9 OverrideFillMode

The **OverrideFillMode** property specifies the object's image filling mode, when displaying a Windows Metafile file. It changes the image's original filling mode without changing the file defined on the **Filename** property. The next table contains valid values for the **OverrideFillMode** property.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	The image will keep its original filling (default).
1 - WideFrame	The image will not be filled.
2 - SolidFill	The image will be filled with the color specified by the **OverrideFillColor** property.
3 - ByBrightness	The image will be filled with the color specified by the **OverrideFillColor** property. However, it will take into account what was the intensity of the original image color.

Example:

```
Sub DrawPicture1_Click()
  ' When clicking the object sets the
  ' Override mode to solid and changes the image filling
  ' color to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.10 OverrideLineColor

When the **EnableOverrideLineColor** property is set to True, the **OverrideLineColor** property will be used to define the color to be used as the picture's color line, instead of the original color. In scripts, use the VBScript's **RGB** method to build the color to be linked to this property. The default value of this property is red (**RGB(255, 0, 0)**). Example:

```
Sub DrawPicture1_Click()
  OverrideLineColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.11 Shadow

Indicates the presence of a shadow effect in the object. If set to True, the object has a shadow, whose coordinates are established by the **ShadowX** and **ShadowY** properties. Otherwise, the object does not have a shadow effect. The default value of this property is False.

4.3.2.7.1.12 ShadowColor

Specifies the object's filling color. This color is used when the **Shadow** property is set to True. In scripts, use the VBScript's **RGB** method to build the color associated to this property. The default value of this property is light gray (**RGB(128, 128, 128)**). Example:

```
Sub Button1_Click()
  ' Changes the background color of the button
  ' to light gray when clicking the object
  ShadowColor = RGB(192, 192, 192)
End Sub
```

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.13 ShadowX

Defines the object shadow's vertical left coordinate, in Himetric units. This shadow is always relative to the object's **X** property. Positive values indicate that the shadow will be on the object's right side, and negative values on the object's left side. The default value of this property is 200.

4.3.2.7.1.14 ShadowY

Defines the object shadow's upper horizontal coordinate, in Himetric units. This shadow is always relative to the object's **Y** property. Positive values indicate the shadow will be under the object, and negative values will be above the object. The default value of this property is 200.

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.15 TransparentColor

When the **TransparentMode** property is set to 1, this property defines what image color will not be drawn, and the image will remain transparent in these areas. In scripts, use the VBScript's **RGB** method to build the color to be associated to this property. The default value of this property is white (**RGB(255, 255, 255)**). Example:

```
Sub DrawPicture1_Click()
  ' Leaves the blue color transparent
  ' when clicking the object
  TransparentMode = 1 ' ByColor
  TransparentColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.16 TransparentMode

The **TransparentMode** property specifies the way the image will handle the transparency effect.

Available options for TransparentMode

OPTION	DESCRIPTION
0 - Disabled	No transparency will be performed.
1 - ByColor	Transparency will use the color set in the **TransparentColor** property.

OPTION	DESCRIPTION
2 - ByPercent	The image will be transparent with its transparency percentage specified in the **TransparentPercent** property.

Example:

```
Sub DrawPicture1_Click()
  ' Leaves the blue color transparent
  ' when clicking the Picture object
  TransparentMode = 1 ' ByColor
  TransparentColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.17 TransparentPercent

When the **TransparentMode** property is set to 2, this property defines how transparent the color will be, varying from 0 (fully transparent) to 100 (fully opaque or solid). Example:

```
Sub DrawPicture1_Click()
  ' Leaves the image transparent when clicking the object
  TransparentMode = 2 ' ByPercent
  TransparentPercent = 50 ' 50% transparent
End Sub
```

NOTE: This property only works when a Picture object is working with metafiles (WMF or EMF).

4.3.2.7.1.18 VerticalFillStyle

Defines the object's vertical filling. This property works with the **VerticalPercentFill** property, which informs what is the percentage of the object will be filled. These two properties allow simulating a level filling in the object.

Available options for VerticalFillStyle

OPTION	DESCRIPTION
0 - FillBottomToTop	Filling percentage will be from bottom to top.
1 - FillTopToBottom	Filling percentage will be from top to bottom.
2 - FillCenterToEdgesV	Filling percentage will be from center to border.

Example:

```
Sub Circle1_OnStartRunning()
  VerticalFillStyle = 2
End Sub
```

4.3.2.7.1.19 VerticalPercentFill

9 Use the **VerticalPercentFill** property to specify what is the percentage of the vertical area of the object will be filled. Acceptable values for this property vary from 0 to 100. This property works with the **VerticalFillStyle** property, which informs how this filling will occur. The default value of this property is 100. Example:

```
Sub Circle1_OnStartRunning()
  VerticalPercentFill = 254
End Sub
```

4.3.2.8 Text, Display, and SetPoint

This section contains information about **events** and **properties** of Text, Display, and SetPoint objects. These objects do not have methods associated to them.

4.3.2.8.1 Events

This section contains information about the events of Text, Display, and SetPoint objects.

4.3.2.8.1.1 Validate

Validate(*Cancel*, *NewValue*)

Occurs after testing the SetPoint limits (see **MinLimit**, **MaxLimit**, and **EnableLimits** properties) and before sending the SetPoint value to the Tag. This event allows the user to validate SetPoint values.

If *Cancel* is True, the attribution operation must be canceled. Otherwise, the value will be sent to the Tag. *NewValue* is the value to validate. The old value can be accessed through the SetPoint's **Value** property. Example:

```
Sub Text1_Validate(Cancel, NewValue)
  ' Displays a MessageBox which asks the user
  ' if the new SetPoint value should be used
  message = "Current value: " & value & vbnewline & _
    "New value: " & NewValue & vbnewline & vbnewline & _
    "Do you accept the new value?"
  If MsgBox (message, vbQuestion + vbYesNo, _
    "Validate Event") = vbNo Then
    Cancel = True
  End If
End Sub
```

4.3.2.8.2 Properties

This section contains information about the properties of Text, Display, and SetPoint objects.

4.3.2.8.2.1 EnableLimits

Checks for limits in the text. If True, and the user inserts a non-number value, or a value outside the limits defined by **MinLimit** and **MaxLimit** properties, an error message is displayed (**IsSetPoint** property must be True). Example:

```
Sub CommandButton1_Click()
  Screen.Item("Text1").EnableLimits = _
    Not(Screen.Item("Text1").EnableLimits)
End Sub
```

4.3.2.8.2.2 Format

A Specifies the type of format to be attributed to the object. It allows changing the way data is presented, without changing the value itself. Users can edit this property manually, or set it through the formatting window. Its usage is similar to format codes on spreadsheets, following the same basic syntax. The following data types are supported:

Data types supported by the Format property

DATA TYPE	DESCRIPTION
Number	Decimal, scientific, hexadecimal, binary, and octal output.
Text	General text.
Boolean	**Boolean** values.
Date/Time	Gregorian calendar.

4.3.2.8.2.3 IsSetPoint

Determines whether the object allows editing the **Value** property. The **Value** property is a **Variant** (it can assume any and every type of value) displayed by the Text object. If True, editing is allowed; otherwise, it is not. It can be viewed when Viewer object is running. Default value is False.

4.3.2.8.2.4 KeepFormatWhenEditing

Allows users to edit object's value, with or without keeping its format. The available options are:

- **0 - kfNever**: in this case, the value is always edited without formatting (default)
- **1 - kfAutomatic**: allows the value to be edited in the formatted value, in case E3 detects it is possible for the formatted text to be interpreted as a value

In case the value is considered as incompatible, the value is edited without formatting.

4.3.2.8.2.5 MaxLimit

Contains the maximum value allowed for the object (the **EnableLimits** property must be True). Example:

```
Sub CommandButton3_Click()
  Screen.Item("Text1").MaxLimit = Screen.Item("Text6").Value
End Sub
```

4.3.2.8.2.6 MinLimit

Contains the minimum value allowed for the object (the **EnableLimits** property must be True). Example:

```
Sub CommandButton2_Click()
  Screen.Item("Text1").MinLimit = Screen.Item("Text5").Value
End Sub
```

4.3.2.8.2.7 Multiline

This property indicates whether the Text has multiple lines (True) or it is a simple text box (False). This can be viewed when Viewer is running. Default value is False. Example:

```
Sub Screen1_OnStartRunning()
  Screen.Item("TextBox1").Multiline = True
End Sub
```

4.3.2.8.2.8 SetPointDataType

Determines the type of SetPoint value sent to the Tag.

Available options for SetPointDataType

OPTION	DESCRIPTION
0 - stCurrentType	Maintains current type (see next).
1 - stChar	8-bit signed integer value.
2 - stByte	8-bit unsigned integer value.
3 - stWord	16-bit unsigned integer value.
4 - stInteger	16-bit signed integer value.
5 - stLong	32-bit signed integer value.
6 - stDWord	16-bit unsigned integer value.
7 - stSingle	32-bit floating point value.
8 - stDouble	64-bit floating point value.
9 - stDateTime	Date and time value.
10 - stString	Text.

When the typed text is sent by the SetPoint, it will first try to convert the value to the configured type (**Word**, **String**, **Double**, etc.). If conversion is not possible, that is, the typed value is not valid for the chosen type, no value is sent (for example, if the user types "-1", and the type is **Byte**). But when property value is **0 - stCurrentType**, the data type sent by the SetPoint comes from the previous value available in the object. In case it is Empty, or Null, no conversion will be performed, and the typed value will be sent as text. Example:

```
Sub Combobox1_Change()
  Screen.Item("Text1").SetPointDataType = CInt(Left(Value, 2))
End Sub
```

4.3.2.8.2.9 StretchText

Resizes the object's type. When enabled, the object automatically resizes its text font size, so that the area occupied by the object always remains the same. Otherwise, if the property is set to False, no resizing will be performed.

4.3.2.8.2.10 TextAlignment

This property specifies the horizontal alignment of the text displayed in the object.

Available options for TextAlignment

OPTION	DESCRIPTION
0 - LeftAlignment	Text horizontal alignment on the left.
1 - CenterAlignment	Text horizontal alignment centered.
2 - RightAlignment	Text horizontal alignment on the right.

4.3.2.8.2.11 TextColor

Specifies the font color in the text. Uses the VBScript's **RGB** method to compose the color to be associated to this property. Default value is black (**RGB(0, 0, 0)**).

4.3.2.8.2.12 TextFont

A Defines the font type to be used in the object. This property cannot be used in Links, and contains the following sub-properties, which can be changed via script:

TextFont sub-properties

NAME	TYPE	DESCRIPTION
Name	String	Font name.
Size	Floating Point	Font size, in points.
Bold	Boolean	Indicates whether text has a bold style.

NAME	TYPE	DESCRIPTION
Italic	Boolean	Indicates whether text has an italic style.
Underline	Boolean	Indicates whether text has an underline style.
Strikethrough	Boolean	Indicates whether text has a strike-through style.
Weight	Integer	Indicates a variation on font's bold effect (the **Bold** sub-property). Values for this sub-property may range from 0 (does not interfere with the bold effect) to 1000.
Charset	Integer	Font character set. See the table of character sets below, for a list of all available values for this sub-property.

Available values for Charset sub-property

VALUE	DESCRIPTION
0	ANSI (*American National Standards Institute*)
1	Default (displays the font's available character set)
2	Symbol
128	Japanese (Shift-JIS)
129	Korean
134	Chinese Simplified (GBK)
136	Chinese Traditional (Big5)
161	Greek
162	Turkish
177	Hebrew
178	Arabic
186	Baltic
204	Russian
222	Thai
238	Central Europe
255	OEM (*Original Equipment Manufacturer*)

4.3.2.8.2.13 Value

The **Value** property is a **Variant** (it can assume any and every type of value) displayed by the object. Usually this property contains a text, because it is automatically filled in when creating a new Text object. The **IsSetPoint** property

determines whether the object allows editing the **Value** property. Example:

```
Sub DrawString1_OnStartRunning()
  ' Reads a tag value and displays the Text
  Dim obj
  Set obj = Application.GetObject("DataServer1.DemoTag1")
  Value = "DemoTag1 value = " & obj.Value
End Sub
```

4.3.2.8.2.14 VertTextAlignment

Determines the vertical alignment of the object text.

Available options for VertTextAlignment

OPTION	DESCRIPTION
0 - TopAlignment	Text vertical alignment on top of the object.
1 - MidAlignment	Text vertical alignment centered in the object.
2 - BottomAlignment	Text vertical alignment at the bottom of the object.

4.3.2.8.2.15 WordWrap

Enables or disables a line break in the text, in case the available area for the text overrides the limits defined by the object. For this property to work, the **Multiline** property must be set to a True.

4.3.2.9 Scale

This section contains information about **properties** of the Scale object. This object does not have events nor methods associated to it.

4.3.2.9.1 Properties

This section contains information about the properties of the Scale object.

4.3.2.9.1.1 BackgroundColor

This property specifies the object's background filling color. This color is used when the **BackgroundStyle** property is set to 1 (opaque) and one of the **VerticalPercentFill** or **HorizontalPercentFill** properties has a value different from 100. Another usage of this color is when the **FillStyle** property is set to values between 2 and 8. This makes the remaining area to use the background color for filling. In scripts, use the VBScript's **RGB** method to build the color to be associated to this property. The default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.9.1.2 BorderColor

This property determines the border color applied on the Scale object. With this property, it is possible to apply a default color, or personalize it by editing the color. The default value of this property is white (**RGB(255, 255, 255)**). Example:

```
Sub Scale1_Click()
  BorderColor = RGB (255, 0, 0)
End Sub
```

4.3.2.9.1.3 BorderStyle

The **BorderStyle** property determines the style of the border applied to the Scale object.

Available options for BorderStyle

OPTION	DESCRIPTION
0 - Normal	Applies a solid border on the Scale's vertical grid (default).
1 - Dash	Applies a dash-line border on the Scale.
2 - Dot	Applies a dotted border on the Scale.
3 - Dashdot	Applies a dash-dot border on the Scale.
4 - Dashdotdot	Applies a dash-dot-dot border on the Scale.
5 - Null	The object has no border.

Example:

```
Sub Scale1_Click()
  BorderStyle = 1
End Sub
```

4.3.2.9.1.4 BorderWidth

This property determines the width (in pixels) of the Scale border. By using this property, it is possible to set the border width, without changing its structure. The default value of this property is 0. Example:

```
Sub CommandButton1_Click()
  Screen.Item("Scale1").BorderWidth = 120
End Sub
```

4.3.2.9.1.5 ForegroundColor

This property specifies the object's foreground filling color. This color is used when the **FillStyle** property is set to 0 (solid) or between 2 and 9. In scripts, use the VBScript's **RGB** method to build the color to be associated to this property. The default value of this property is blue (**RGB(0, 0, 255)**). Example:

```
Sub Button1_Click()
  ' Changes the background color of the button
  ' to green when clicking the object
  ForegroundColor = RGB(0, 255, 0)
End Sub
```

4.3.2.9.1.6 Format

A Specifies the format type to be attributed to the object. It allows changing the way data is presented without changing their values. Users can edit this property manually, or set it through the Format window. Its usage is similar to formatting codes on spreadsheets, following the same basic syntax. The following data types are supported.

Data types supported by Format

DATATYPE	DESCRIPTION
Number	Decimal, scientific, hexadecimal, binary, and octal output.
Text	Text in general.
Boolean	**Boolean** values.
Date/Time	Gregorian calendar.

4.3.2.9.1.7 GradientStyle

Specifies the object's gradient fill style. Used only when the **FillStyle** property is set to 8 (Gradient). Gradients range from **ForegroundColor** to **BackgroundColor** properties.

Available options for GradientStyle

OPTION	DESCRIPTION
0 - LeftToRight	Vertical gradient from left to right.
1 - RightToLeft	Vertical gradient from right to left.
2 - VerFromCenter	Vertical gradient from center to border.
3 - VerToCenter	Vertical gradient from border to center.
4 - BottomUp	Horizontal gradient from bottom to top.
5 - TopDown	Horizontal gradient from top to bottom.
6 - HorzFromCenter	Gradient from center to border.
7 - HorzToCenter	Gradient from border to center.
8 - DiagUpRight	Diagonal top gradient with foreground color on the right (default).
9 - DiagUpLeft	Diagonal top gradient with foreground color on the left.
10 - DiagUpFromCenter	Diagonal top gradient from center to border.
11 - DiagUpToCenter	Diagonal top gradient from border to center.
12 - DiagDownLeft	Diagonal bottom gradient with foreground color on the left.

OPTION	DESCRIPTION
13 - DiagDownRight	Diagonal bottom gradient with foreground color on the right.
14 - DiagDownFromCenter	Diagonal bottom gradient from center to border.
15 - DiagDownToCenter	Diagonal bottom gradient from border to center.
16 - SpotSouthEast	Gradient with foreground color from right lower corner.
17 - SpotSouthWest	Gradient with foreground color from left lower corner.
18 - SpotNorthWest	Gradient with foreground color from left upper corner.
19 - SpotNorthEast	Gradient with foreground color from right upper corner.
20 - SpotFromCenter	Gradient with background color from center to border.
21 - SpotToCenter	Gradient with background color from border to center.

IMPORTANT: Too many objects being displayed simultaneously with gradients leads to poor performance when updating the Screen. Using pictures may solve the problem.

Example:

```
Sub Button1_Click()
  ' Objeto gets a gradient
  FillStyle = 8 ' GradientFill
  GradientStyle = 0 ' LeftToRight
End Sub
```

4.3.2.9.1.8 LineColor

Determines the line color for Scale's ticks and minor ticks. To determine the legend color with object's numbers, use the **TextColor** property. Default value is black (**RGB(0, 0, 0)**).

4.3.2.9.1.9 MaximumValue

9 This property determines the maximum value reached by the Scale. The default value of this property is 100. Example:

```
Sub CommandButton_Click()
  ' When clicking the button, opens a MessageBox
  ' indicating the maximum value of the property
  MsgBox CSTr(Screen.Item("Scale1").MaximumValue)
End Sub
```

4.3.2.9.1.10 MinimumValue

This property determines the minimum value reached by the Scale. The default value of this property is 0. Example:

```
Sub CommandButton1_Click()
  ' When clicking the button, opens a MessageBox
  ' indicating the minimum value of the property
  MsgBox _
    CSTr(Application.GetObject("Data.Scale1").MinimumValue)
End Sub
```

4.3.2.9.1.11 MinorTicks

Determines the number of minor ticks on the Scale. The default value is 3. Example:

```
Sub CommandButton1_Click()
  ' Displays the total amount of minor ticks on the scale
  MsgBox CStr(Screen.Item("Scale1").MinorTicks)
End Sub
```

4.3.2.9.1.12 MinorTicksPercentSize

This property determines the size of the marks subdividing each Scale's measurement. The default value of this property is 10. Example:

```
Sub CommandButton1_Click()
  MsgBox CStr(Screen.Item("Scale1").MinorTicksPercentSize)
End Sub
```

4.3.2.9.1.13 ScaleAlignment

This property determines the type of Scale's alignment:

- **0 - RightSide**: to the right (default value)
- **1 - LeftSide**: to the left

Example:

```
Sub CommandButton1_Click()
  Screen.Item("Scale1").ScaleAlignment = 1
End Sub
```

4.3.2.9.1.14 ShowText

Determines whether text is displayed in the Scale's legend. If True, text is displayed. Otherwise, the object displays only scale lines and minor ticks. Default value is True.

4.3.2.9.1.15 StretchText

Applies stretching to the object's text (whenever height or width vary, the text box follows the change). If True, the text stretches to accommodate to the object's new dimensions. Otherwise, text maintains its original settings. Default value is False.

4.3.2.9.1.16 TextAlignment

The **TextAlignment** property determines the object's text alignment.

Available option for TextAlignment

OPTION	DESCRIPTION
0 - leftAlignment	Left text alignment (default).
1 - centerAlignment	Centered text alignment.
2 - rightAlignment	Right text alignment.

4.3.2.9.1.17 TextColor

Determines the font color applied to the Scale's legend numbers. To determine the line color with ticks and minor ticks, use the **LineColor** property. Default value is black (**RGB(0, 0, 0)**).

4.3.2.9.1.18 TextFont

A Determines the font to be applied to the Scale. This property cannot be used in Links. See the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

4.3.2.9.1.19 Ticks

9 Determines the number of ticks on the scale. Default value is 5.

4.3.2.9.1.20 TicksPercentSize

9 Determines the size of the lines dividing the Scale object. The size of the Scale's default divider line will be smaller or bigger, according to this value. The default value of this property is 20.

4.3.2.10 Dynamic Move

This section contains information about the **properties** of the Dynamic Move object. This object does not have events nor methods associated to it.

4.3.2.10.1 Properties

This section contains information about the properties of the Dynamic Move object.

4.3.2.10.1.1 Detents

The **Detents** property determines the number of steps for the object movement. Example:

```
Sub CommandButton1_Click()
  MsgBox Screen.Item("DynamicRotate1").Detents
End Sub
```

4.3.2.10.1.2 EnableOverrideLineColor

Enables or disables the object to override its original line color by the color defined in the **OverrideLineColor** property. Otherwise, each object in the group will present its original line color. Default value is False.

4.3.2.10.1.3 EnableSlider

This property enables a slider in object's movement. If True, enables movement. Otherwise, movement is disabled.

4.3.2.10.1.4 OverrideFillColor

When **OverrideFillMode** property is set to 2 or 3, the **OverrideFillColor** property is used to define the object's fill color, instead of its original color. Use the VBScript's **RGB** method to compose the color to be associated to this property. Default value is red (**RGB(255, 0, 0)**). Example:

```
Sub DrawGroup1_Click()
  ' When clicking the object sets the
  ' Override mode to solid and changes the
  ' filling color to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

4.3.2.10.1.5 OverrideFillMode

The **OverrideFillMode** property specifies the filling mode of the moving objects. It changes the image's original filling mode, without changing the objects' original filling settings.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	Original filling of the object.

OPTION	DESCRIPTION
1 - WireFrame	Objects will not be filled, only their borders will be drawn.
2 - SolidFill	Filling of the objects contained in a group will be solid with the color specified in the **OverrideFillColor** property.
3 - ByBrightness	Filling of the objects contained in a group will be solid with the color specified in the **OverrideFillColor** property, however it will take into account the brightness of the original filling color of every object.

Example:

```
Sub DrawGroup1_Click()
  ' When clicking the object sets the
  ' Override mode to solid and changes the
  ' filling color of the image to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

4.3.2.10.1.6 OverrideLineColor

When **EnableOverrideLineColor** property is set to True, **OverrideLineColor** property is used to define the color to be used on object lines, instead of their original color. Use the VBScript's **RGB** method to compose the color to be associated to this property. Default value is red (**RGB(255, 0, 0)**). Example:

```
Sub Group1_Click()
  OverrideLineColor = RGB(255, 0, 0)
End Sub
```

4.3.2.10.1.7 RangeMax

9 Determines maximum range of the object's linear sliding.

4.3.2.10.1.8 RangeMin

9 Determines minimum range of the object's linear sliding.

4.3.2.10.1.9 Value

This is the Slider's initial value. It must be a value between **RangeMax** and **RangeMin** properties.

4.3.2.11 Dynamic Rotate

This section contains information about the **properties** of the Dynamic Rotate object. This object doe not have events nor methods associated to it.

4.3.2.11.1 Properties

This section contains information about the properties of the Dynamic Rotate object.

4.3.2.11.1.1 Detents

This property determines the number of steps of the object's movement.

4.3.2.11.1.2 EnableOverrideLineColor

This property enables or disables the object to overwrite the original color of the image line by the color defined in the **OverrideLineColor** property. If the **EnableOverrideLineColor** property is enabled, the original color of the object's line is changed by the color set in the **OverrideLineColor** property. Otherwise, the object displays its original color. The default value of this property is False.

4.3.2.11.1.3 EnableSlider

The **EnableSlider** property enables or disables the slider in the object's movement.

4.3.2.11.1.4 OverrideFillColor

When the **OverrideFillMode** property is set to 2 or 3, the **OverrideFillColor** property is used to define object's filling color, instead of its original color. In scripts, use the VBScript's **RGB** function to create the color to be associated to this property. The default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.11.1.5 OverrideFillMode

The **OverrideFillMode** property specifies the moving objects' filling mode. It changes the original filling mode of the image without changing the objects' original filling settings.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	Original filling of the object.
1 - WireFrame	Objects are not filled, they only draw their borders.

OPTION	DESCRIPTION
2 - SolidFill	The filling of the objects is solid with the color specified in the **OverrideFillColor** property.
3 - ByBrightness	The filling of the objects is solid with the color specified in the **OverrideFillColor** property, however, the brightness of the original color of each object is preserved.

4.3.2.11.1.6 OverrideLineColor

When the **EnableOverrideLineColor** property is set to True, the **OverrideLineColor** property is used to define the color to be used in the line of the moving object, instead of the original color. In scripts, use the VBScript's **RGB** function to create the color to be associated to this property. The default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.11.1.7 RangeMax

This property determines the maximum range reached by the object's rotational movement.

4.3.2.11.1.8 RangeMin

This property determines the minimum range reached by the object's rotational movement.

4.3.2.11.1.9 RotationAngle

This property determines the object movement's rotation angle. Example:

```
Sub CommandButton1_DbClick()
  Screen.Item("DynamicMove1").RotationAngle = 180
End Sub
```

4.3.2.11.1.10 RotationDirection

The **RotationDirection** property determines the orientation of object movement's rotation.

Available options for RotationDirection

OPTION	DESCRIPTION
0 - Clockwise	Sets the rotation angle to the right of the object.
1 - CounterClockWise	Sets the rotation angle to the left of the object.

Example:

```
Sub CommandButton1_Click()
  ' Directs the rotation angle to the right of the object
  Screen.Item("DynamicRotate1").RotationDirection = 1
End Sub
```

4.3.2.11.1.11 Value

This is the movement's initial value. It must be a value between the **RangeMax** and **RangeMin** properties.

4.3.2.12 Microsoft Forms

This section contains information about common **events** and **properties** of Microsoft Forms objects. These objects do not have common methods associated to them.

4.3.2.12.1 Common Events

This section contains information about common events of Microsoft Forms objects.

4.3.2.12.1.1 BeforeDragOver

BeforeDragOver(*Index*, *Cancel*, *Data*, *X*, *Y*, *DragState*, *Effect*, *Shift*)

It occurs when there is a drag-and-drop action on the object. Use this event to monitor if the mouse entered, left or stood over a target object. This event is triggered when users move the mouse or press and release any button. Mouse pointer's position indicates the object generating the event. Users can specify the mouse pointer status by checking the *DragState* variable.

Many controls (objects) do not support drag-and-drop operations while the *Cancel* variable is False, which is the default behavior. This means that the control rejects any attempt of dragging and dropping some object on itself, and therefore it does not trigger the **BeforeDropOrPaste** event. The Text Box and the Combo Box are exceptions, because they accept drag-and-drop operations even when the *Cancel* variable is False.

Available parameters of the BeforeDragOver event

PARAMETER	DESCRIPTION
Index	Indicates, in multi-page objects, the index of the page affected by the operation that generated the event. For other objects, it is ignored.

PARAMETER	DESCRIPTION
Cancel	Event status. By default it is False and indicates that the target object addresses the event and not the main application.
Data	Data being dragged to the target object.
X, Y	Mouse position in points within the target object. *X* is measured from the left side of the object; *Y* is measured from the top.
DragState	Indicates mouse condition when the event is generated: • **0 - fmDragStateEnter**: mouse is within the object's range • **1 - fmDragStateLeave**: mouse is outside the object's range • **2 - fmDragStateOver**: mouse is at a new position, but still within the object's range
Effect	Indicates the actions supported by the target object, that is, the drag effect on the mentioned object: • **0 - fmDropEffectNone**: target object does not accept copying or moving from any origin • **1 - fmDropEffectCopy**: target object allows copying from any origin to itself • **2 - fmDropEffectMove**: target object allows moving from any origin to itself • **3 - fmDropEffectCopyOrMove**: target object allows copying or moving from any origin to itself
Shift	Integer whose sum indicates SHIFT, CTRL, and ALT key status: • **1**: SHIFT key pressed • **2**: CTRL key pressed • **4**: ALT key pressed For example, a value of 5 indicates that SHIFT and ALT keys were pressed (1 + 4).

4.3.2.12.1.2 BeforeDropOrPaste

BeforeDropOrPaste(*Index*, *Cancel*, *Ctrl*, *Action*, *Data*, *X*, *Y*, *Effect*, *Shift*)

Triggered immediately before a drag-and-drop operation. Normally, it occurs right after a **BeforeDragOver** event.

Available parameters of the BeforeDropOrPaste event

NAME	DESCRIPTION
Index	Indicates, in multi-page objects, the index of the page affected by the operation that generated the event. For other objects, it is ignored.
Cancel	Event status. By default it is False and indicates that the target object addresses the event and not the main application.
Ctrl	Target object.
Data	Data being dragged to the target object.
Action	Indicates the result, based on keyboard settings, of a pending drag-and-drop operation: • **2 - fmActionPaste**: pastes selected object on target object • **3 - fmActionDragDrop**: indicates that users dragged selected object from its origin and dropped it on the target object
X, Y	Mouse position in points within the target object. *X* is measured from the left side of the object; *Y* is measured from the top.
DragState	Indicates mouse condition when the event is generated: • **0 - fmDragStateEnter**: mouse is within object's range • **1 - fmDragStateLeave**: mouse is outside object's range • **2 - fmDragStateOver**: mouse is at a new position, but still within the object's range
Effect	Indicates the actions supported by the target object, that is, the drag effect on the mentioned object: • **0 - fmDropEffectNone**: target object does not accept copying or moving from any origin • **1 - fmDropEffectCopy**: target object allows copying from any origin to itself • **2 - fmDropEffectMove**: target object allows moving from any origin to itself • **3 - fmDropEffectCopyOrMove**: target object allows copying or moving from any origin to itself

NAME	DESCRIPTION
Shift	Integer whose sum indicates SHIFT, CTRL, and ALT keys status: • **1**: SHIFT key pressed • **2**: CTRL key pressed • **4**: ALT key pressed For example, a value of 5 indicates that SHIFT and ALT keys were pressed (1 + 4).

4.3.2.12.1.3 Change

Change()

It occurs when the value of the object's **Value** property changes. Here are some examples of actions triggering the **Change** event:

- Clicking a Check Box, an Option Button, or an Increase-Decrease Button
- Clicking or selecting words on a Selection List or a Text Editor
- Selecting different tabs on a dialog box
- Moving the scroll bar in a Scroll Bar object
- Clicking the arrows of an Increase-Decrease Button
- Selecting different pages in a Multi-Page object

4.3.2.12.1.4 KeyPress

KeyPress(*KeyAscii*)

This event occurs when the object has keyboard focus and user presses a key corresponding to a character that can be displayed on Screen (an ANSI key, with its code indicated in the *KeyAscii* variable). That is, this event occurs when any of the following keys are pressed:

- Any keyboard character that can be printed
- The CTRL key combined with any standard alphabet character
- The CTRL key combined with any special character
- The BACKSPACE key
- The ESC key

This event does not occur in the following conditions:

- When pressing the TAB key
- When pressing the ENTER key

- When pressing the DEL key (this is not an ANSI key)
- When pressing keyboard arrow keys
- When a key changes focus from one object to another

While a user is pressing a key that produces an ANSI code, the object repeatedly receives the **KeyDown** and the **KeyPress** events. When the user releases the key, the **KeyUp** event occurs. To monitor keyboard physical status, or handle keys not recognized by the **KeyPress** event (such as function keys, browsing keys, etc.), use the **KeyDown** and **KeyUp** events.

4.3.2.12.1.5 OnError

OnError(*Number*, *Description*, *SCode*, *Source*, *HelpFile*, *HelpContext*, *CancelDisplay*)

Generated by an object's internal error. If this event is not handled, E3 then displays a generic error message.

Available parameters of the OnError event

PARAMETER	DESCRIPTION
Number	Integer identifying the error.
Description	**String** with error description.
SCode	Integer with the error code of the OLE subsystem (not used).
Source	**String** with the object that generated the error.
HelpFile	**String** with name and path of the help file.
HelpContext	Context number of the help topic referring to the error (Integer).
CancelDisplay	**Boolean** indicating whether the error should be displayed on a message box.

4.3.2.12.2 Common Properties

This section contains information about common properties of Microsoft Forms objects.

NOTE: E3 uses the Himetric system to define coordinates and widths. In this system, each logical unit is the equivalent of a thousandth of a centimeter, that is, every 1,000 units correspond to one centimeter. Thus, this is the standard adopted in the description of E3 properties, when applicable.

4.3.2.12.2.1 BackColor

Specifies the object's background color. In scripts, use the VBScript's **RGB** method to compose the color to be associated to this property. Default value for Combo, List, and Text is white (**RGB(255, 255, 255)**). For all other objects, it is beige (**RGB(236, 233, 216)**).

4.3.2.12.2.2 ForeColor

Specifies the object's foreground color. In scripts, use the VBScript's **RGB** method to compose the color to be associated to this property. The default value of this property for all MS Forms objects is black (**RGB(0, 0, 0)**).

4.3.2.12.2.3 MouseIcon

This property sets an image to the mouse pointer, whenever it moves over an object. This property is valid only when the **MousePointer** property is set to **99 - fmMousePointerCustom**.

An image file can be selected to be used as mouse pointer in two different ways. Via Property Window (.cur or .ico extensions), or via scripts, by using the **LoadPicture** method to specify the name and the path of the file with the customized icon (.cur extension only). Example:

```
Sub CommandButton1_Click()
  ' Setting the 99 - fmMousePointerCustom item to the property
  ' so that it accepts mouse icon customization
  Screen.Item("CheckBox1").MousePointer = 99
  Screen.Item("CheckBox1").MouseIcon = LoadPicture("C:\a.cur")
End Sub
```

4.3.2.12.2.4 MousePointer

Specifies the type of the mouse pointer displayed when the user moves it over an object. The available options are:

Available options for MousePointer

OPTION	DESCRIPTION
0 - fmMousePointerDefault	Default pointer. The image is determined by the object.
1 - fmMousePointerArrow	Arrow.
2 - fmMousePointerCross	Arrow shaped pointer.
3 - fmMousePointerBeam	I-beam shaped pointer.
6 - fmMousePointerSizeNesw	North-East, South-West double arrow shaped pointer.
7 - fmMousePointerSizeNS	North-South double arrow shaped pointer.

OPTION	DESCRIPTION
8 - fmMousePointerNWse	North-West, South-East double arrow shaped pointer.
9 - fmMousePointerWE	West-East double arrow shaped pointer.
10 - fmMousePointerUpArrow	Upper arrow.
11 - MousePointerStarHourGlassring	Hourglass.
12 - fmMousePointerHelpNoDrop	A Not symbol (a circle with a diagonal line) on the upper side of the dragged object. Indicates and invalid drop destination.
13 - fmMousePointerAppStarting	Hourglass shaped pointer.
14 - fmMousePointerHelp	Question mark pointer.
15 - fmMousePointerSizeAll	Resizes the whole cursor (arrows pointing to North, South, East, and West).
99 - fmMousePointerCustom	Uses the icon specified in the **MouseIcon** property.

Use this property to indicate a change in functionality as the mouse pointer slides over Screen objects. For example, an hourglass shaped pointer (option **11**) is useful to indicate that the user should wait for the process or operation to be finished. Some icons may vary, depending on the system configuration, like icons associated to working area themes. The default value of this property is **0 - fmMousePointerDefault**.

4.3.2.12.3 Checkbox

This section contains information about **properties** of the Checkbox object. This object does not have events nor methods associated to it.

4.3.2.12.3.1 Properties

This section contains information about the properties of the Checkbox.

Accelerator

A Defines or gets the object's accelerator key. This accelerator key is a shortcut key that, when used together with the ALT key, gives focus to the object. Default value is an empty **String**.

Alignment

Specifies the object's alignment on the Screen, relative to its legend. The available options are:

- **0 - fmAlignmentLeft**: aligns the legend to the left of the object
- **1 - fmAligmentRight**: aligns the legend to the right of the object

AutoSize

Adjusts text width, in case its available area is larger than the object's size. For Checkbox and Option Button objects, if this property is True, the text will be resized to fit the object's current size.

BackStyle

Defines the object's background style. The available options are:

- **0 - fmBackStyleTransparent**: defines the object as transparent, that is, no background will be drawn
- **1 - fmBackStyleOpaque**: defines the object as opaque, that is, the background will be drawn (default value)

NOTE: This property does no affect bitmap transparency. Users should use an image editor such as Paintbrush, for example, to save a bitmap as transparent. Not all ActiveX objects support transparent bitmaps.

Caption

Defines the text to be displayed on the object.

Font

Determines the object's font. This property cannot be used in scripts or Links, and it is configured exclusively via E3 Studio.

GroupName

Creates a mutually exclusive group of objects.

NOTE: This property is not used in E3, and it is kept for compatibility reasons with Microsoft Forms standard specification.

Locked

Enables or disables object edition. If True, edition is not allowed; otherwise, it is. The value configured for the **Enabled** property influences the **Locked** property behavior. For further details, see the **Enabled** property. Default value is False.

Picture

Specifies the picture (bitmap) set to the object. An image file can be selected in two ways: via Properties List; or via scripts, by using the **LoadPicture** method to specify the path and the name of the file containing the picture. To remove the picture, click the value of the **Picture** property and press the DEL key. The BACKSPACE key does not delete the picture. Example:

```
Sub CommandButton1_Click()
  Screen.Item("CheckBox1").Picture = LoadPicture("C:\tab.gif")
End Sub
```

PicturePosition

Specifies the object's picture position, relative to its legend. The available options for this property are:

Available options for PicturePosition

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on the left of its legend. The legend is aligned to the upper side of the picture.
1 - fmPicturePositionLeftCenter	Picture appears on the left of its legend. The legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on the left of its legend. The legend is aligned to the lower part of the picture.
3 - fmPicturePositionRightTop	Picture appears on the right of its legend. The legend is aligned to the upper side of the picture.
4 - fmPicturePositionRightCenter	Picture appears on the right of its legend. The legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on the right of its legend. The legend is aligned to the lower part of the picture.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. The legend is aligned to the left side of the picture.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. The legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. The legend is aligned to the right side of the picture.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. The legend is aligned to the left side of the picture.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. The legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. The legend is aligned to the right side of the picture.

SpecialEffect

Specifies the object's appearance. The available options are the following:

Available options for SpecialEffect

OPTION	DESCRIPTION
0 - fmButtonEffectFlat	The object appears to be flat and has a raised border, a color change, or both.
2 - fmButtonlEffectSunken	The object has a shadow on the upper left side and is raised on the lower right side, as if it is sunken on the Screen.

TextAlign

Specifies how text is aligned in the object. The available options are:

- **1 - fmTextAlignLeft**: aligns the text to the left side of the object
- **2 - fmTextAlignCenter**: centers the text to the right side of the object
- **3 - fmTextAlignRight**: aligns the text to the right side of the object

TripleState

Determines up to three value status for the object. If True, the user can choose from three status options: **False**, **True**, or **Null**. The **Null** state is displayed as a shaded button. Otherwise, the user can only choose from **True** or **False** values. Default value is **False**.

Value

Indicates the object's initial value. Its behavior is **Boolean**. If True, its initial status is checked, otherwise it will start unchecked. Default value is False.

WordWrap

Enables or disables line breaks in the text, in case the available area for the text overrides the boundaries determined in the object.

4.3.2.12.4 Option Button

This section contains information about **properties** of the Option Button object. This object does not have events nor methods associated to it.

4.3.2.12.4.1 Properties

This section contains information about the properties of the Option Button.

Accelerator

Defines or gets the object's accelerator key. This accelerator key is a shortcut key that, when used together with the ALT key, gives focus to the object. Default value is an empty **String**.

Alignment

Specifies the object's alignment on the Screen, relative to its legend. The available options are:

- **0 - fmAlignmentLeft**: aligns the legend to the left of the object
- **1 - fmAligmentRight**: aligns the legend to the right of the object

AutoSize

Adjusts text width, in case its available area is larger than the object's size. For Checkbox and Option Button objects, if this property is True, the text will be resized to fit the object's current size.

BackStyle

Defines the object's background style. The available options are:

- **0 - fmBackStyleTransparent**: defines the object as transparent, that is, no background will be drawn
- **1 - fmBackStyleOpaque**: defines the object as opaque, that is, the background will be drawn (default value)

NOTE: This property does no affect bitmap transparency. Users should use an image editor such as Paintbrush, for example, to save a bitmap as transparent. Not all ActiveX objects support transparent bitmaps.

Caption

A Defines the text to be displayed on the object.

Font

A Determines the object's font. This property cannot be used in scripts or Links, and it is configured exclusively via E3 Studio.

GroupName

A Creates a mutually exclusive group of objects.

NOTE: This property is not used in E3, and it is kept for compatibility reasons with Microsoft Forms standard specification.

Locked

Enables or disables object edition. If True, edition is not allowed; otherwise, it is. The value configured for the **Enabled** property influences the **Locked** property behavior. For further details, see the **Enabled** property. Default value is False.

Picture

Specifies the picture (bitmap) set to the object. An image file can be selected in two ways: via Properties List; or via scripts, by using the **LoadPicture** method to specify the path and the name of the file containing the picture. To remove the picture, click the value of the **Picture** property and press the DEL key. The BACKSPACE key does not delete the picture. Example:

```
Sub CommandButton1_Click()
  Screen.Item("OptionButton1").Picture = LoadPicture("C:
\tab.gif")
End Sub
```

PicturePosition

Specifies the object's picture position, relative to its legend. The available options for this property are:

Available options for PicturePosition

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on the left of its legend. The legend is aligned to the upper side of the picture.
1 - fmPicturePositionLeftCenter	Picture appears on the left of its legend. The legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on the left of its legend. The legend is aligned to the lower part of the picture.
3 - fmPicturePositionRightTop	Picture appears on the right of its legend. The legend is aligned to the upper side of the picture.
4 - fmPicturePositionRightCenter	Picture appears on the right of its legend. The legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on the right of its legend. The legend is aligned to the lower part of the picture.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. The legend is aligned to the left side of the picture.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. The legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. The legend is aligned to the right side of the picture.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. The legend is aligned to the left side of the picture.

OPTION	DESCRIPTION
10 - fmPicturePositionBelowCenter	Picture appears below the legend. The legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. The legend is aligned to the right side of the picture.

SpecialEffect

Specifies the object's appearance. The available options are the following:

Available options for SpecialEffect

OPTION	DESCRIPTION
0 - fmButtonEffectFlat	The object appears to be flat and has a raised border, a color change, or both.
2 - fmButtonlEffectSunken	The object has a shadow on the upper left side and is raised on the lower right side, as if it is sunken on the Screen.

TextAlign

Specifies how text is aligned in the object. The available options are:

- **1 - fmTextAlignLeft**: aligns the text to the left side of the object
- **2 - fmTextAlignCenter**: centers the text to the right side of the object
- **3 - fmTextAlignRight**: aligns the text to the right side of the object

TripleState

Determines up to three value status for the object. If True, the user can choose from three status options: **False**, **True**, or **Null**. The **Null** state is displayed as a shaded button. Otherwise, the user can only choose from **True** or **False** values. Default value is **False**.

Value

Indicates the object's initial value. Its behavior is **Boolean**. If True, its initial status is checked, otherwise it will start unchecked. Default value is False.

WordWrap

Enables or disables line breaks in the text, in case the available area for the text overrides the boundaries determined in the object.

4.3.2.12.5 Combo

This section contains information about **events**, **methods**, and **properties** of the Combo object.

4.3.2.12.5.1 Events

This section contains information about the events of the Combo object.

DropButtonClick

DropButtonClick()

It occurs when a Combo is displayed or hidden after clicking the object.

4.3.2.12.5.2 Methods

This section contains information about the methods of the Combo object.

AddItem

AddItem([*pvargItem*[, *pvargIndex*]])

Adds items to a Combo. *pvargItem* is a **String** containing the text to be added to the Combo. If omitted, an empty **String** will be added. *pvargIndex* is the index of the Combo's text. If omitted, *pvargItem* is added as the last item on the Combo. Example:

```
Sub CommandButton1_Click()
  EntryCount = EntryCount + 1
  ComboBox1.AddItem(EntryCount & " - Selection")
End Sub
```

Clear

Clear()

Clears the object's text. Example:

```
Sub ClearTextButton_Click()
  ComboBox1.Clear()
End Sub
```

Copy

Copy()

Copies the previously selected text to the Clipboard. Use the **Paste** method to paste this text into the indicated place. Example:

```
Sub CommandButton1_Click()
  Screen.Item("ComboBox1").Copy()
End Sub
```

Cut

Cut()

Cuts the previously selected text to the Clipboard. Use the **Paste** method to paste this text into the indicated place. Example:

```
Sub CommandButton1_Click()
  Screen.Item("ComboBox1").Cut()
```

```
End Sub
```

DropDown

DropDown()

This method opens the Combo's list of items. This is the same as clicking, at run time, the object's right side button. Example:

```
Sub CommandButton1_Click()
  Dim ComboBox1
  ComboBox1.AddItem "Pineapple"
  ComboBox1.AddItem "Strawberry"
  ComboBox1.AddItem "Grape"
  ComboBox1.AddItem "Orange"
  ComboBox1.DropDown()
End Sub
```

Paste

Paste()

Inserts the Clipboard content into the object. Example:

```
Sub CommandButton1_Click()
  Screen.Item("ComboBox1").Paste()
End Sub
```

RemoveItem

RemoveItem(*pvargIndex*)

Removes items from a Combo. This method has the *pvargIndex* parameter specifying the item to be removed. The first item is 0, the second one is 1, and so on. Example:

```
Sub CommandButton2_Click()
  ComboBox1.SetFocus
  ' Check if the list contains the selected data
  If ComboBox1.ListCount >= 1 Then
    ' If there is no selection, choose the last item.
    If ComboBox1.ListIndex = -1 Then
      ComboBox1.ListIndex = ComboBox1.ListCount - 1
    End If
    ComboBox1.RemoveItem(ComboBox1.ListIndex)
  End If
End Sub
```

4.3.2.12.5.3 Properties

This section contains information about the properties of the Combo object.

AutoSize

◩ Adjusts text width, in case its available area is larger than the object's size. For the Combo Box object, if this property is True, the text will be resized to fit the object's current size.

AutoTab

Enables or disables the object's automatic tabbing. If True, tabbing is automatic. Otherwise, it is not. After users have typed the maximum number of characters in an object (using the **MaxLength** property), focus moves automatically to the next object in the tab order, whenever these characters are reached. For example, for a Combo Box to always display stored data with five characters, users can use the **MaxLength** property to specify the maximum number of characters to be typed in the object, and the **AutoTab** property to automatically move to the next object after users have typed five characters.

AutoWordSelect

Enables or disables the object's automatic word selection. If True, the whole word is selected, plus the next white space. Otherwise, just the indicated character is selected.

BackStyle

Defines the object's background style. The available options are:

- **0 - fmBackStyleTransparent**: defines the object as transparent, that is, no background will be drawn
- **1 - fmBackStyleOpaque**: defines the object as opaque, that is, the background will be drawn (default value)

NOTE: This property does no affect bitmap transparency. Users should use an image editor such as Paintbrush, for example, to save a bitmap as transparent. Not all ActiveX objects support transparent bitmaps.

BorderColor

Defines the object's border color. Thus, it is possible to use the default color, or personalize it while editing. To apply this property, the **BorderStyle** property must be set to **1 - fmBorderStyleSingle**. Default value is black (**RGB(0, 0, 0)**).

BorderStyle

Defines the object's border style. The available options are:

- **0 - fmBorderStyleNone**: no border
- **1 - fmBorderStyleSingle**: single border

BoundColumn

Establishes the column where data is stored on the list. For example, if each row has eight items, and the **BoundColumn** property is set to 3, the application stores information on the third column of the currently selected row. If value is set to 0 (zero), it is passed to the object's **ListIndex** property. If value is 1 or higher, the indicated data is attributed to the column referring to the value specified in this

property. Columns start at 1.

NOTE: This property has no effect on E3, and it is kept for compatibility reasons with Microsoft Forms standard specification.

CanPaste

Specifies whether the Clipboard contains data supported by the object. If True, the object can receive information pasted from the Clipboard. However, if data in this area is in a format the object does not support, this property is set to False. This property is only available at run time.

Column

Specifies the object's row and column. If only the column value is specified, the **Column** property reads or writes the column specified on the object's current row. For example, `MyComboBox.Column(3)` reads or writes the object's third column. This property is only available at run time.

ColumnCount

Specifies the number of columns of the object. If it is set to 0, no columns will be displayed. If set to -1, all available columns are displayed. Default value is 1.

ColumnHeads

Enables or disables displaying column titles in the object. If True, the title is displayed. Otherwise, it is not. Default value is False.

ColumnWidths

Specifies the object's column width, in points. If value is -1 or blank, width is calculated on the column (minimum width in a calculated column is 72 points, or 1 inch). If value is 0 (zero), column is hidden. To produce narrower columns, users must specify a width in this property, or use one of the values on the next table.

Available options for ColumnWidths

OPTION	DESCRIPTION
90;72;90	First column is 90 points (1.25 inches); second column is 72 points (1 inch); third column is 90 points.
6 cm;0;6 cm	First column is 6 centimeters; second column is hidden; third column is 6 centimeters. As part of the third column is visible, an horizontal scroll bar is visible.
1.5 in;0;2.5 in	First column is 1.5 inches, second column is hidden, and the third column is 2.5 inches.

OPTION	DESCRIPTION
2 in;;2 in	First column is 2 inches, second column is 1 inch (default), and the third column is 2 inches. As only half of the third column is visible, an horizontal scroll bar is visible.
(Empty)	All three columns are the same width (1.33 inches). The default value of this property is empty (E3 will use the system's default value).

CurTargetX

9 Returns the horizontal insertion position of a text in the object. This position is measured in Himetric units. Users can use either **CurTargetX** or **CurX** properties to move the insertion point in a text as users pass over the object's content. When users move the insertion point to another row in the text, the **CurTargetX** property specifies the most appropriate position for the text's insertion point. **CurX** property is defined in this value, if the text row is bigger than the value of **CurTargetX**. Otherwise, **CurX** property is defined as the text row ending. This property is only available at run time.

NOTE: This property has no effect in E3, and is kept for compatibility reasons with the Microsoft Forms standard specification.

CurX

9 Specifies current horizontal text insertion position in the object. This property is applied to an object with many rows, that is, whenever **Multiline** property is enabled. The return value is valid when the object has focus. Users can use either **Multiline** or **CurX** properties to place the text insertion point as users pass over the object's content. When users move the insertion point to another row in the text, **CurTargetX** property specifies the most appropriate position for the text's insertion point. **CurX** property is defined in this value, if the text row is bigger than the value of **CurTargetX**. Otherwise, **CurX** property is defined as the text row ending. This property is only available at run time.

DragBehavior

Enables or disables drag-and-drop operations of a text in the object's content. The available options for this property are:

- **0 - fmDragBehaviorDisabled**: does not allow dragging and dropping a text in the object's content
- **1 - fmDragBehaviorEnabled**: allows dragging and dropping a text in the object's content

The default value of this property is **0 - fmDragBehaviorDisabled**.

NOTE: The **DragBehavior** property has no effect if the **Style** property is set to **2 - fmStyleDropDownList**.

DropButtonStyle

Specifies the symbol displayed in the Combo button. The available options for this property are:

Available options for DropButtonStyle

OPTION	DESCRIPTION
0 - fmDropButtonStylePlain	Displays a plain button, without a symbol.
1 - fmDropButtonStyleArrow	Displays a down arrow (default).
2 - fmDropButtonStyleEllipsis	Displays an ellipsis.
3 - fmDropButtonStyleReduce	Displays an horizontal line with an underscore character.

The default value of this property is **1 - fmDropButtonStyleArrow**.

EnterFieldBehavior

Controls how text content is selected in the edition area when the TAB key is pressed on the object, and not when the object receives focus as the result of a **SetFocus** method. The available options are:

- **0 - fmEnterFieldBehaviorSelectAll**: selects the whole text content when the TAB key is pressed in the object
- **1 - fmEnterFieldBehaviorRecallSelection**: selection remains unchanged

The default value of this property is **0 - fmEnterFieldBehaviorSelectAll**.

Font

A Determines the object's font. It cannot be used in scripts or Links, and it is configured exclusively via E3 Studio.

HideSelection

Specifies whether the selected text is still highlighted when the object does not have focus anymore. If True, the selected text only remains highlighted in case the object has the focus. Otherwise, the object will always appear highlighted, no matter if it has the focus. Default value is True.

IMEMode

The **IMEMode** property specifies the object's IME (*Input Method Editor*) mode.

NOTE: This property only applies to applications written in Eastern languages (Chinese Simplified and Traditional, Korean, and Japanese) and it is ignored in other applications. It was kept for compatibility reasons with Microsoft Forms standard specification.

The available options are the following:

Available options for IMEMode

OPTION	DESCRIPTION
0 - fmIMEModeNoControl	No IME control (default).
1 - fmIMEModeOn	IME mode on.
2 - fmIMEModeOff	IME mode off. English mode.
3 - fmIMEModeDisable	IME mode off. Users cannot activate IME mode via keyboard.
4 - fmIMEModeHiragana	IME mode on with Hiragana mode (Full Width).
5 - fmIMEModeKatakanaFull	IME mode on with Katakana mode (Full Width).
6 - fmIMEModeKatakana	IME mode on with Katakana mode (Half Width).
7 - fmIMEModeAlphaFull	IME mode on with Alphanumeric mode (Full Width).
8 - fmIMEModeAlpha	IME mode on with Alphanumeric mode (Half Width).
9 - fmIMEModeHangulFull	IME mode on with Hangul mode (Full Width).
10 - fmIMEModeHangul	IME mode on with Hangul mode (Half Width).
11 - fmIMEModeHanziFull	IME mode on with Hanzi mode (Full Width).
12 - fmIMEModeHanzi	IME mode on with Hanzi mode (Half Width).

LineCount

9 Returns the object's row number. This property is only available at run time.

List

v Returns or defines column and row entries on the object's list. Column and row numbering starts at zero; this means that the number of the first column and of the first row on the list are zero, the second column and row are 1, and so on. This property is only available at run time.

ListCount

9 Returns the number of items in the object's list. This property is only available at run time.

ListIndex

9 Identifies the item currently selected on the list (index). Values for this property go from -1 to the total amount of rows on a list minus one (that is, **ListCount** - 1). When no row is selected, **ListIndex** returns -1. When the user selects a row in a Combo, the system defines the value of **ListIndex** property. The list's first row value of this property is 0, the second row value is 1, and so on. This property is only available at run time.

ListRows

9 Determines the maximum amount of rows on the object's list. Default value is 8.

ListStyle

Determines the object's list style. The available options for this property are:

- **0 - fmListStylePlain**: list with background items in plain style
- **1 - fmListStyleOption**: displays option buttons or combo boxes for a list with several options. When the user selects one item on the group, the option button associated to this item is then selected, and the option buttons of the other group items are unchecked

The default value of this property is **0 - fmListStylePlain**.

ListWidth

9 Determines the width of the object's list. Default value is 0.

Locked

◩ Enables or disables object edition. If True, edition is not allowed; otherwise, it is. The value configured for the **Enabled** property influences the behavior of the **Locked** property. For further details, see the **Enabled** property. Default value is False.

MatchEntry

Searches for a text entry that matches data in the object. When a text instance is found, the row with the text is highlighted, and column content is displayed. The available options are the following:

- **0 - fmMatchEntryFirstLetter**: search for a text entry that matches the first character typed in the object. If the same character is repeatedly typed, moves to the next entry type starting with that character, and so on
- **1 - fmMatchEntryComplete**: as every character is typed, the object searches for a text entry that matches the typed character
- **2 - fmMatchEntryNone**: does not perform a search in the object

The default value of this property is **1 - fmMatchEntryComplete**.

MatchFound

◩ Indicates whether the text that users have typed in the object matches any entry on the list. If True, the **Value** property content matches one of the records on the list. Otherwise, the content of the **Value** property does not match any of the records on the list (default). This property is available only at run time, and does not apply if the **MatchEntry** property is set to 2. Default value is False.

MatchRequired

◩ Specifies whether the typed text should match or not the items in the Combo. If True, users cannot leave the box until the text inserted matches an item on the object. Otherwise, the inserted text on the Combo can be different from all existing data on it.

MaxLength

9 Determines the maximum number of characters in the object. If set to 0 (zero), there will be no character limits on the object.

SelectionMargin

◩ Enables or disables the object's selection margin. If True, the text will be selected when clicking the object's margin. Otherwise, the text will not be selected when clicking the margin.

NOTE: If the **SelectionMargin** property is set to True when the object is being printed,the selection margin will also be printed.

SelLength

9 Returns the number of selected characters in the object. This property is only available at run time.

SelStart

9 Indicates the starting point of the selected text, or the insertion point if no text is selected. This property is only available at run time.

SelText

A Returns the text selected in the object. This property is only available at run time.

ShowDropButtonWhen

Specifies when the drop button (object's browsing key) is displayed. The available options are:

- **0 - fmShowDropButtonWhenNever**: does not display a drop button
- **1 - fmShowDropButtonWhenFocus**: only display the drop button when the object has the focus

- **2 - fmShowDropButtonWhenAlways**: always displays a drop button

SpecialEffect

Specifies the object's appearance. The available options are:

Available options for SpecialEffect

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object looks flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on the upper left side, and a shadow on the right lower side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on the left upper side, and is raised on the right lower side. The object and its border looks sunken on the Screen.
3 - fmSpecialEffectEtched	Border looks etched around the object edges.
6 - fmSpecialEffectBump	Object has a ledge on the right lower side, and looks flat on the left upper side.

Style

Determines the object's style. The available options are:

- **0 - fmStyleDropDownCombo**: the object behaves as a drop down combo. Users may type a value on the edition area, or select a value from the drop down list (default)
- **2 - fmStyleDropDownList**: the object behaves as a list. Users must choose a value from a list

Text

Returns the text of the selected option. This property is only available at run time.

TextAlign

Specifies how text is aligned inside the object. The available options are:

- **1 - fmTextAlignLeft**: aligns the text to the left of the object
- **2 - fmTextAlignCenter**: aligns the text to the center of the object
- **3 - fmTextAlignRight**: aligns the text to the right of the object

TextColumn

Identifies a column in the object. The values for this property range from -1 to the number of columns on the list. The value of **TextColumn** property for the first column is 1, for the second column is 2, and so on. When the **TextColumn** property

is set to 0, values of the **ListIndex** property will be displayed. When the **TextColumn** property is set to -1, the first column with **ColumnWidths** property value greater than 0 will be displayed.

TextLength

9 Returns the number of characters typed in the object. This property is only available at run time.

TopIndex

9 Defines or returns the item in the combo that is on top of the list. This property returns -1 if the list is empty or hidden.

Value

v This is the value in the **BoundColumn** property of the currently selected rows. For a Combo Box, changing the contents of the **Value** property does not change the value of the **BoundColumn** property. To add or delete entries in a Combo, users can use either **AddItem** or **RemoveItem** methods.

4.3.2.12.6 Command Button

This section contains information about **events** and **properties** of the Command Button object. This object does not have methods associated to it.

4.3.2.12.6.1 Events

This section contains information about the events of the Command Button object.

MouseMove

MouseMove()

This event occurs when the mouse pointer is moved over a Command Button.

4.3.2.12.6.2 Properties

This section contains information about the properties of the Command Button object.

Accelerator

A Defines or gets the object's accelerator key. This accelerator key is a shortcut key that, when used with the ALT key, gives focus to the object. The default value of this property is an empty **String**.

AutoSize

◩ Adjusts text width, in case its available area is larger than the object's size. For the Command Button object, if this property is True, the text will be resized to fit the

current object's size. The text content is cut when it exceeds the object's area.

BackStyle

Defines the object's background style. The available options are:

- **0 - fmBackStyleTransparent**: defines the object as transparent, that is, no object background will be drawn
- **1 - fmBackStyleOpaque**: defines the object as opaque, that is, the background will be drawn (default value)

NOTE: This property does no affect bitmap transparency. Users should use an image editor such as Paintbrush, for example, to save a bitmap as transparent. Not all ActiveX objects support transparent bitmaps.

Caption

Defines the text to be displayed in the object.

Font

This property is used to determine the object's font. This property cannot be used on scripts or Links, it must be configured only in Studio.

Locked

Enables or disables object edition. If True, edition is not allowed; otherwise, it is. The value configured for the **Enabled** property influences the **Locked** property behavior. For further details, see the **Enabled** property. Default value is False.

Picture

Specifies the object's picture (bitmap). An image file can be selected in two ways: via Properties List; or via scripts, by using the **LoadPicture** function to specify the path and the name of the file containing the picture. To remove the picture, click the value of the **Picture** property and press the DEL key. The BACKSPACE key does not delete the picture.

PicturePosition

Specifies the position of the object's picture, relative to its legend. The available options for this property are:

Available options for PicturePosition

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on the left of its legend. The legend is aligned to the upper side of the picture.
1 - fmPicturePositionLeftCenter	Picture appears on the left of its legend. The legend is centered relative to the picture.

OPTION	DESCRIPTION
2 - fmPicturePositionLeftBottom	Picture appears on the left of its legend. The legend is aligned to the lower part of the picture.
3 - fmPicturePositionRightTop	Picture appears on the right of its legend. The legend is aligned to the upper side of the picture.
4 - fmPicturePositionRightCenter	Picture appears on the right of its legend. The legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on the right of its legend. The legend is aligned to the lower part of the picture.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. The legend is aligned to the left side of the picture.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. The legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. The legend is aligned to the right side of the picture.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. The legend is aligned to the left side of the picture.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. The legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. The legend is aligned to the right side of the picture.

TakeFocusOnClick

◩ Specifies whether the object gets the focus when clicked. If True, it does. Otherwise, the object does not get the focus when clicked.

WordWrap

◩ Enables or disables line breaks in the text, in case the text's available area overrides the limits determined in the object.

4.3.2.12.7 Label

This section contains information about **properties** of the Label object. This object does not have events nor methods associated to it.

4.3.2.12.7.1 Properties

This section contains information about the properties of the Label object.

Accelerator

Defines or get the object's accelerator key. This accelerator key is a shortcut key that, when used with the ALT key, gives focus to the object. The default value of this property is an empty **String**.

AutoSize

Adjusts text width, in case its available area is larger than the object's size. For the Label object, when this property is set to True, text is resized to match the object's current size.

BackStyle

Defines the object's background style. The available options are:

- **0 - fmBackStyleTransparent**: defines the object as transparent, that is, no object background will be drawn
- **1 - fmBackStyleOpaque (valor padrão)**: defines the object as opaque, that is, the background will be drawn (default value)

NOTE: This property does no affect bitmap transparency. Users should use an image editor such as Paintbrush, for example, to save a bitmap as transparent. Not all ActiveX objects support transparent bitmaps.

BorderColor

Defines the object's border color. Thus, it is possible to use the default color, or personalize it while editing. To apply this property, the **BorderStyle** property must be set to **1 - fmBorderStyleSingle**. Default value is black (**RGB(0, 0, 0)**).

BorderStyle

Defines the object's border style. The available options are:

- **0 - fmBorderStyleNone**: no border
- **1 - fmBorderStyleSingle**: single border

Caption

Defines the object's text.

Font

This property is used to determine the object's font. This property cannot be used on scripts or Links, it must be configured only in Studio.

Picture

Specifies the object's picture (bitmap). An image file can be selected in two ways: via Properties List; or via scripts, by using the **LoadPicture** function to specify the

path and the name of the file containing the picture. To remove the picture, click the value of **Picture** property and press the DEL key. The BACKSPACE key does not delete the picture.

PicturePosition

Specifies the position of the object's picture, relative to its legend. The available options for this property are:

Available options for PicturePosition

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on the left of its legend. The legend is aligned to the upper side of the picture.
1 - fmPicturePositionLeftCenter	Picture appears on the left of its legend. The legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on the left of its legend. The legend is aligned to the lower part of the picture.
3 - fmPicturePositionRightTop	Picture appears on the right of its legend. The legend is aligned to the upper side of the picture.
4 - fmPicturePositionRightCenter	Picture appears on the right of its legend. The legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on the right of its legend. The legend is aligned to the lower part of the picture.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. The legend is aligned to the left side of the picture.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. The legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. The legend is aligned to the right side of the picture.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. The legend is aligned to the left side of the picture.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. The legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. The legend is aligned to the right side of the picture.

SpecialEffect

Specifies the object's appearance. The available options are:

Available options for SpecialEffect

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object looks flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on the upper left side, and a shadow on the right lower side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on the left upper side, and is raised on the right lower side. The object and its border looks sunken on the Screen.
3 - fmSpecialEffectEtched	Border looks etched around the edges of the object.
6 - fmSpecialEffectBump	Object has a ledge on the right lower side, and looks flat on the left upper side.

TextAlign

Specifies how text is aligned inside the object. The available options are:

- **1 - fmTextAlignLeft**: aligns the text to the left of the object
- **2 - fmTextAlignCenter**: aligns the text to the center of the object
- **3 - fmTextAlignRight**: aligns the text to the right of the object

WordWrap

Enables or disables line breaks in the text, in case the text's available area overrides the limits determined in the object. For this property to work, the **Multiline** property must be True.

4.3.2.12.8 List

This section contains information about **methods** and **properties** of the List object. This object does not have events associated to it.

4.3.2.12.8.1 Methods

This section contains information about the methods of the List object.

AddItem

AddItem([*pvargItem*[, *pvargIndex*]])

Adds items to a List. *pvargItem* is a **String** containing the text to be added to the List. If omitted, an empty **String** will be added. *pvargIndex* is the index of the List text. If omitted, *pvargItem* is added as the last item on the List. Example:

```
Sub CommandButton1_Click()
  EntryCount = EntryCount + 1
```

```
  ListBox1.AddItem(EntryCount & " - Selection")
End Sub
```

Clear

Clear()

Clears the object's text. Example:

```
Sub ClearTextButton_Click()
  ListBox1.Clear()
End Sub
```

RemoveItem

RemoveItem(*pvargIndex*)

Removes items from a List. This method has the *pvargIndex* parameter, which specifies the item to be excluded. The first item number is 0, the second item number is 1, and so on. Example:

```
Sub CommandButton2_Click()
  List1.SetFocus
  ' Checks if the list has selected items
  If List1.ListCount >= 1 Then
    ' If there is no selection, chooses the last item of the
list.
    If List1.ListIndex = -1 Then
      List1.ListIndex = List1.ListCount - 1
    End If
    List1.RemoveItem(List1.ListIndex)
  End If
End Sub
```

4.3.2.12.8.2 Properties

This section contains information about the properties of the List object.

BorderColor

Defines the object's border color. Thus, it is possible to use the default color, or personalize it while editing. To apply this property, the **BorderStyle** property must be set to **1 - fmBorderStyleSingle**. Default value is black (**RGB(0, 0, 0)**).

BorderStyle

Defines the object's border style. The available options are:

- **0 - fmBorderStyleNone**: no border
- **1 - fmBorderStyleSingle**: single border

BoundColumn

Establishes the column where data is stored on the List. For example, if each row has eight items, and the **BoundColumn** property is set to 3, the application stores

information on the third column of the currently selected row. If value is set to 0, it is then passed to the object's **ListIndex** property. If value is set to 1 or higher, the indicated data is attributed to the column referring to the value specified in the property. Columns start at 1.

NOTE: This property has no effect on E3, and it is kept for compatibility reasons with Microsoft Forms standard specification.

Column

Specifies the object's row and column. If only the column value is specified, the **Column** property reads or writes the column specified on the object's current row. For example, `MyListBox.Column(3)` reads or writes the object's third column. This property is only available at run time.

ColumnCount

Specifies the number of columns of the object. If it is set to 0, no columns will be displayed. If set to -1, all available columns will appear. Default value is 1.

ColumnHeads

Enables or disables displaying column titles in the object. If True, the title is displayed. Otherwise, it is not. Default value is False.

ColumnWidths

Specifies the object's column width, in points. If value is -1 or blank, width is calculated on the column (minimum width in a calculated column is 72 points, or 1 inch). If value is 0, column is hidden. To produce narrower columns, users should specify a width in the property, or use one of the values on the next table.

Available options for ColumnWidths

OPTION	DESCRIPTION
90;72;90	First column is 90 points (1.25 inches); second column is 72 points (1 inch); third column is 90 points.
6 cm;0;6 cm	First column is 6 centimeters; second column is hidden; third column is 6 centimeters. As part of the third column is visible, an horizontal scroll bar is visible.
1.5 in;0;2.5 in	First column is 1.5 inches, second column is hidden, and the third column is 2.5 inches.
2 in;;2 in	First column is 2 inches, second column is 1 inch (default), and the third column is 2 inches. As only half of the third column is visible, an horizontal scroll bar is visible.

OPTION	DESCRIPTION
(Empty)	All three columns are the same width (1.33 inches). The default value of this property is empty (E3 will use the system's default value).

Font

A Determines the object's font. It cannot be used in scripts or Links, and it is configured exclusively via E3 Studio.

IMEMode

The **IMEMode** property specifies the object's IME (*Input Method Editor*) mode.

NOTE: This property only applies to applications written in Eastern languages (Chinese Simplified and Traditional, Korean, and Japanese), and it is ignored in other applications. It was kept for compatibility reasons with Microsoft Forms standard specification.

The available options are the following:

Available options for IMEMode

OPTION	DESCRIPTION
0 - fmIMEModeNoControl	No IME control (default).
1 - fmIMEModeOn	IME mode on.
2 - fmIMEModeOff	IME mode off. English mode.
3 - fmIMEModeDisable	IME mode off. Users cannot activate IME mode via keyboard.
4 - fmIMEModeHiragana	IME mode on with Hiragana mode (Full Width).
5 - fmIMEModeKatakanaFull	IME mode on with Katakana mode (Full Width).
6 - fmIMEModeKatakana	IME mode on with Katakana mode (Half Width).
7 - fmIMEModeAlphaFull	IME mode on with Alphanumeric mode (Full Width).
8 - fmIMEModeAlpha	IME mode on with Alphanumeric mode (Half Width).
9 - fmIMEModeHangulFull	IME mode on with Hangul mode (Full Width).
10 - fmIMEModeHangul	IME mode on with Hangul mode (Half Width).
11 - fmIMEModeHanziFull	IME mode on with Hanzi mode (Full Width).
12 - fmIMEModeHanzi	IME mode on with Hanzi mode (Half Width).

IntegralHeight

Adjusts height in text editing area, in case its available area exceeds the object's size. If True, its height is adjusted to fit the object's current size, thus displaying the whole text. Otherwise, the size of text's editing area remains the same. If a text is larger than the available space, it will not be displayed in the object.

List

Returns or defines column and row entries on the object's list. Column and row numbering starts at zero; it means that the number of the first column and of the first row on the list are zero, the second column and row are 1, and so on. This property is only available at run time.

ListCount

Returns the number of items in the object's list. This property is only available at run time.

ListIndex

Identifies the item currently selected on the list (index). The values of this property go from -1 to the total amount of rows on a list minus one (that is, **ListCount** - 1). When no row is selected, **ListIndex** returns -1. When users select a row on a List, the application defines the value of **ListIndex** property. The List's first row value of this property is 0, the second row value is 1, and so on. This property is only available at run time.

ListStyle

Determines the object's list style. The available options for this property are:

- **0 - fmListStylePlain**: a list with background items in plain style
- **1 - fmListStyleOption**: displays option buttons or combo boxes for a list with several options. When users select one group item, the option button associated to this item is selected, and the option buttons of the other group items are unchecked

The default value of this property is **0 - fmListStylePlain**.

NOTE: The option **1 - fmListStyleOption** can only be enabled if the **MultiSelect** property is set to **1 - fmMultiselectMulti**.

Locked

Enables or disables object edition. If True, edition is not allowed; otherwise, it is. The value configured for the **Enabled** property influences the behavior of the **Locked** property. For further details, see the **Enabled** property. Default value is False

MatchEntry

Searches for a text entry that matches data in the object. When a text instance is found, the row with that text is highlighted, and column content is displayed. The available options are the following:

- **0 - fmMatchEntryFirstLetter**: searches for a text entry that matches the first character typed in the object. If the same character is repeatedly typed, moves to the next entry type that starts with that character, and so on
- **1 - fmMatchEntryComplete**: as every character is typed, the object searches for a text entry that matches the typed characters
- **2 - fmMatchEntryNone**: does not perform a search in the object

The default value of this property is **1 - fmMatchEntryComplete**.

MultiSelect

Indicates whether the object allows multiple selections. The available options for this property are:

- **0 - fmMultiSelectSingle**: only one item can be selected
- **1 - fmMultiSelectMulti**: allows selecting one item by pressing the space bar, or by clicking the mouse, thus checking or unchecking an item on the list
- **2 - fmMultiSelectExtended**: allows selecting one item by pressing the SHIFT key, by clicking the mouse, or by pressing the SHIFT key and one of the arrow keys, extending the selection to the current item. Pressing the CTRL key and clicking the mouse, checks or unchecks the item

The default value of this property is **0 - fmMultiSelectSingle**.

Selected

Selects or deselects an item, and checks whether an item is selected, when **Multiline** property is True. To check whether the item is selected or not, the item index must be passed, and the property returns its selection status. So, it is possible to determine what items are selected when users select more than one. This property is only available at run time. When multiple selections are not used, it is recommended to use the **Value** or the **ListIndex** properties.

SpecialEffect

Specifies the object's appearance. The available options are:

Available options for SpecialEffect

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object looks flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on the upper left side, and a shadow on the right lower side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on the left upper side, and is raised on the right lower side. The object and its border looks sunken on the Screen.
3 - fmSpecialEffectEtched	Border looks etched around the edges of the object.
6 - fmSpecialEffectBump	Object has a ledge on the right lower side, and looks flat on the left upper side.

Text

A Returns the text of the selected option. This property is only available at run time.

TextAlign

Specifies how text is aligned inside the object. The available options are:

- **1 - fmTextAlignLeft**: aligns the text to the left of the object
- **2 - fmTextAlignCenter**: aligns the text to the center of the object
- **3 - fmTextAlignRight**: aligns the text to the right of the object

TextColumn

Identifies a column in the object. The values for this property range from -1 to the number of columns on the list. **TextColumn** property value for the first column is 1, for the second column is 2, and so on. When the **TextColumn** property is set to 0, values of the **ListIndex** property will be displayed. When the **TextColumn** property is set to -1, the first column with **ColumnWidths** property value greater than 0 will be displayed.

TopIndex

9 Defines or returns the item on the List which is on top of the list. This property returns -1 if the List is empty or hidden.

Value

This is the value of the **BoundColumn** property of the currently selected rows. This property has no effect in E3, and is kept for compatibility reasons with Microsoft Forms standard specification.

4.3.2.12.9 Toggle Button

This section contains information about **events** and **properties** of the Toggle Button. This object does not have methods associated to it.

4.3.2.12.9.1 Events

This section contains information about the events of the Toggle Button object.

MouseMove

MouseMove()

This event occurs when the mouse pointer is moved over a Toggle Button object.

4.3.2.12.9.2 Properties

This section contains information about the properties of the Toggle Button object.

Accelerator

A Defines or gets the object's accelerator key. This accelerator key is a shortcut key that, when used with the ALT key, gives focus to the object. The default value of this property is an empty **String**.

Alignment

Specifies the object's alignment on the Screen, relative to its legend. The available options are:

- **0 - fmAlignmentLeft**: aligns the legend to the left of the object
- **1 - fmAligmentRight**: aligns the legend to the right of the object

AutoSize

Adjusts text width, in case its available area is larger than the object's size. For the Toggle Button object, when this property is set to True, the text is resized to match the object's current size.

BackStyle

Defines the object's background style. The available options are:

- **0 - fmBackStyleTransparent**: defines the object as transparent, that is, no object background will be drawn
- **1 - fmBackStyleOpaque**: defines the object as opaque, that is, the background will be drawn (default value)

NOTE: This property does no affect bitmap transparency. Users should use an image editor such as Paintbrush, for example, to save a bitmap as transparent. Not all ActiveX objects support transparent bitmaps.

Caption

A Defines the text to be displayed in the object.

Font

A This property is used to determine the object's font. This property cannot be used on scripts or Links, it must be configured only in Studio.

GroupName

A Creates a mutually exclusive group of objects. This property is only available at run time.

NOTE: This property is not used in E3, and it is kept for compatibility reasons with Microsoft Forms standard specification.

Locked

◩ Enables or disables object edition. If True, edition is not allowed; otherwise, it is. The value configured for the **Enabled** property influences the **Locked** property behavior. For further details, see the **Enabled** property. Default value is False.

Picture

Specifies the object's picture (bitmap). An image file can be selected in two ways: via Properties List; or via scripts, by using the **LoadPicture** function to specify the path and the name of the file containing the picture. To remove the picture, click the value of **Picture** property and press the DEL key. The BACKSPACE key does not delete the picture.

PicturePosition

Specifies the position of the object's picture, relative to its legend. The available options for this property are:

Available options for PicturePosition

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on the left of its legend. The legend is aligned to the upper side of the picture.
1 - fmPicturePositionLeftCenter	Picture appears on the left of its legend. The legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on the left of its legend. The legend is aligned to the lower part of the picture.

OPTION	DESCRIPTION
3 - fmPicturePositionRightTop	Picture appears on the right of its legend. The legend is aligned to the upper side of the picture.
4 - fmPicturePositionRightCenter	Picture appears on the right of its legend. The legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on the right of its legend. The legend is aligned to the lower part of the picture.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. The legend is aligned to the left side of the picture.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. The legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. The legend is aligned to the right side of the picture.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. The legend is aligned to the left side of the picture.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. The legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. The legend is aligned to the right side of the picture.

SpecialEffect

Specifies the object's appearance. The available options are:

Available options for SpecialEffect

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object looks flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on the upper left side, and a shadow on the right lower side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on the left upper side, and is raised on the right lower side. The object and its border looks sunken on the Screen.
3 - fmSpecialEffectEtched	Border looks etched around the edges of the object.
6 - fmSpecialEffectBump	Object has a ledge on the right lower side, and looks flat on the left upper side.

TextAlign

Specifies how text is aligned inside the object. The available options are:

- **1 - fmTextAlignLeft**: aligns the text to the left of the object
- **2 - fmTextAlignCenter**: aligns the text to the center of the object
- **3 - fmTextAlignRight**: aligns the text to the right of the object

TripleState

Determines up to three value status for the object. If True, users can choose from three status options: **False**, **True**, or **Null**. The **Null** status is displayed as a shaded button. Otherwise, users can only choose from **True** or **False** values. Default value is **False**.

Value

Indicates the object's initial value. It has a **Boolean** behavior. If True, the object starts checked; otherwise, its initial state is unchecked. The default value of this property is False.

WordWrap

Enables or disables line breaks in the text, in case the available area for the text overrides the object limits. For this property to work, **Multiline** property must be True.

4.3.2.12.10 Text

This section contains information about **events**, **methods**, and **properties** of the Text object.

4.3.2.12.10.1 Events

This section contains information about the events of the Text object.

DropButtonClick

DropButtonClick()

This event occurs when the list of options appear or disappear after clicking this object.

4.3.2.12.10.2 Methods

This section contains information about the methods of the Text object.

Copy

Copy()

Copies to the Clipboard a previously selected text. Use the **Paste** method to paste the text in another place. Example:

```
Sub CommandButton1_Click()
  Screen.Item("TextBox1").Copy()
End Sub
```

Cut

Cut()

Cuts to the Clipboard a previously selected text. Use the **Paste** method to paste the text in another place. Example:

```
Sub CommandButton1_Click()
  Screen.Item("TextBox1").Cut()
End Sub
```

Paste

Paste()

Inserts the Clipboard content into the Text. Example:

```
Sub CommandButton1_Click()
  Screen.Item("TextBox1").Paste()
End Sub
```

4.3.2.12.10.3 Properties

This section contains information about the properties of the Text object.

AutoSize

◩ The **AutoSize** property adjusts text width, in case its available area exceeds the object's size. For the Text object, when this property is set to True, the text width is resized to the same object's width. The default value of this property is False.

NOTE: It is advisable to avoid using the **AutoSize** property with a Text that also uses the **Multiline** and **WordWrap** properties. When users type into a Text with these properties set to True, it will automatically resize itself as a long and narrow text box, with a width of one character and a size of one text row.

AutoTab

◩ Enables or disables the object's automatic tabbing. If True, tab is automatic. Otherwise, it is not. After users have typed the maximum number of characters in an object (using the **MaxLength** property), the focus moves automatically to the next object in the tab order, whenever these characters are reached. For example, for a Text object to always display stored data with five characters, users can use the **MaxLength** property to specify the maximum number of characters to be typed in the object, and the **AutoTab** property to automatically move to the next object after users have typed five characters.

AutoWordSelect

Enables or disables the object's automatic word selection. If True, the whole word is selected, plus the next white space. Otherwise, just the indicated character is selected.

BackStyle

Defines the object's background style. The available options are:

- **0 - fmBackStyleTransparent**: defines the object as transparent, that is, no background will be drawn
- **1 - fmBackStyleOpaque**: defines the object as opaque, that is, the background will be drawn (default value)

NOTE: This property does no affect bitmap transparency. Users should use an image editor such as Paintbrush, for example, to save a bitmap as transparent. Not all ActiveX objects support transparent bitmaps.

BorderColor

Defines the object's border color. Thus, it is possible to use the default color, or personalize it while editing. To apply this property, the **BorderStyle** property must be set to **1 - fmBorderStyleSingle**. Default value is black (**RGB(0, 0, 0)**).

BorderStyle

Defines the object's border style. The available options are:

- **0 - fmBorderStyleNone**: no border
- **1 - fmBorderStyleSingle**: single border

CanPaste

Specifies whether the Clipboard contains data supported by the object. If True, the object can receive information pasted from the Clipboard. However, if data in the Clipboard is in a format the object does not support, this property is False. This property is only available at run time.

CurLine

9 Specifies the current object's row, that is, the row that contains the text insertion point. The number of the first row is 0. The default value of this property is 0.

CurTargetX

9 Returns the horizontal insertion position of a text in the object. This position is measured in Himetric units (0,0001 meters). Users can use **CurTargetX** and **CurX** to move the insertion point of a text while moving through the object's content. When moving the insertion point to another row, the **CurTargetX** property specifies the most indicated position for the desired text insertion point. The **CurX** property is

defined with this value, if the text row is greater than the value of **CurTargetX**. Otherwise, the **CurX** property is defined as the end of the text row. This property is only accessible at run time.

CurX

Specifies current horizontal text insertion position in the object. This property is applied to an object with many rows, that is, whenever **Multiline** property is enabled. The return value is valid when the object has the focus. Users can use either **Multiline** or **CurX** properties to place the text insertion point as users pass over the object's content. When users move the insertion point to another row in the text, **CurTargetX** property specifies the most appropriate position for the text's insertion point. **CurX** property is defined in this value, if the text row is greater than the value of **CurTargetX**. Otherwise, **CurX** property is defined as the text row end. This property is only available at run time.

DragBehavior

Enables or disables a drag-and-drop operation of a text in the object's content. The available options for this property are:

- **0 - fmDragBehaviorDisabled**: does not allow dragging and dropping a text in the object's content
- **1 - fmDragBehaviorEnabled**: allows dragging and dropping a text in the object's content

The default value of this property is **0 - fmDragBehaviorDisabled**

NOTE: The **DragBehavior** property has no effect if the **Style** property is set to **2 - fmStyleDropDownList**.

EnterFieldBehavior

Controls how text content is selected in the edition area when TAB key is pressed on the object, and not when the object receives the focus as the result of the **SetFocus** method. The available options are:

- **0 - fmEnterFieldBehaviorSelectAll (default)**: selects all the content of the text when the TAB key is pressed on the object
- **1 - fmEnterFieldBehaviorRecallSelection**: the selection remains untouched

EnterKeyBehavior

Defines the ENTER key behavior in the object. If True, whenever users press the ENTER key, a new line is created in the object's text edition area. Otherwise, whenever users press the ENTER key, the focus goes to the next object in the tab order. This also happens if the **Multiline** property is set to False, regardless of **EnterKeyBehavior** property value.

The combination CTRL + ENTER also depends on the value of the **Multiline** property. If this property is set to True, whenever these two keys are pressed together a new line is created in the object's text edition area, regardless of the **EnterKeyBehavior** property value. If this property is False, these keys will have no effect on the text.

Font

A Determines object's font. It cannot be used in scripts or Links, and it is configured exclusively via Studio.

HideSelection

Specifies whether the selected text is still highlighted when the object does not have focus anymore. If True, the selected text only remains highlighted in case the object has focus. Otherwise, the object will always appear highlighted, regardless of the presence of focus in the object. Default value is True.

IMEMode

The **IMEMode** property specifies the object's IME (*Input Method Editor*) mode.

NOTE: This property only applies to applications written in Eastern languages (Chinese Simplified and Traditional, Korean, and Japanese), and it is ignored in other applications. It was kept for compatibility reasons with Microsoft Forms standard specification.

The available options are the following:

Available options for IMEMode

OPTION	DESCRIPTION
0 - fmIMEModeNoControl	No IME control (default).
1 - fmIMEModeOn	IME mode on.
2 - fmIMEModeOff	IME mode off. English mode.
3 - fmIMEModeDisable	IME mode off. Users cannot activate IME mode via keyboard.
4 - fmIMEModeHiragana	IME mode on with Hiragana mode (Full Width).
5 - fmIMEModeKatakanaFull	IME mode on with Katakana mode (Full Width).
6 - fmIMEModeKatakana	IME mode on with Katakana mode (Half Width).
7 - fmIMEModeAlphaFull	IME mode on with Alphanumeric mode (Full Width).
8 - fmIMEModeAlpha	IME mode on with Alphanumeric mode (Half Width).
9 - fmIMEModeHangulFull	IME mode on with Hangul mode (Full Width).
10 - fmIMEModeHangul	IME mode on with Hangul mode (Half Width).

OPTION	DESCRIPTION
11 - fmIMEModeHanziFull	IME mode on with Hanzi mode (Full Width).
12 - fmIMEModeHanzi	IME mode on with Hanzi mode (Half Width).

IntegralHeight

This property adjusts the height in text edition area, in case its available area exceeds object's size. If True, its height is adjusted to fit the object's current size, thus displaying the whole text. Otherwise, the size of text edition area remains the same. If a text is larger than the available space, it will not be displayed in the object.

LineCount

9 The **LineCount** property returns the number of lines of this object. This property is only available at run time.

Locked

Enables or disables object edition. If True, edition is not allowed; otherwise, it is. The value configured for the **Enabled** property influences the behavior of the **Locked** property. For further details, see the **Enabled** property. Default value is False.

MaxLength

9 Determines the maximum number of characters in the object. If set to 0, there will be no limit for characters in the object.

Multiline

This property indicates whether the text has multiple lines (True) or is a simple text box (False). It can be viewed when Viewer is running. Default value is False.

PasswordChar

A Converts the object's text to a special character configured in this property. Use this property to protect confidential information, such as passwords or security codes. **PasswordChar** property's value is a character (usually an asterisk) that appears in the object, instead of the current characters the user types. If the character is not specified, this control then displays the characters actually typed.

ScrollBars

Specifies whether this object has vertical or horizontal scroll bars, or even both. The available options are:

- **0 - fmScrollBarNone**: no scroll bars are displayed
- **1 - fmScrollBarHorizontal**: a horizontal scroll bar is displayed
- **2 - fmScrollBarVertical**: a vertical scroll bar is displayed

Default value is **0 - fmScrollBarNone**.

SelectionMargin

◩ Enables or disables the object's selection margin. If True, the text will be selected when users click the object's margin. Otherwise, the text will not be selected when users click the margin.

NOTE: If the **SelectionMargin** property is set to True when the object is being printed,the selection margin will also be printed.

SelLength

9 Returns the number of selected characters in the object. This property is only available at run time.

SelStart

9 Indicates the starting point of the selected text, or the insertion point if no text is selected. This property is only available at run time.

SelText

A Returns the text selected in the object. This property is only available at run time.

SpecialEffect

Specifies the object's appearance. The available options are:

Available options for SpecialEffect

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object looks flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on the upper left side, and a shadow on the right lower side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on the left upper side, and is raised on the right lower side. The object and its border looks sunken on the Screen.
3 - fmSpecialEffectEtched	Border looks etched around the edges of the object.
6 - fmSpecialEffectBump	Object has a ledge on the right lower side, and looks flat on the left upper side.

TabKeyBehavior

◩ Determines whether tab order is allowed at edition area. If True, whenever the TAB key is pressed, a space character is inserted at edition area. Otherwise, when the

TAB key is pressed, the focus goes to the next object on the tab order.

Text

Returns the text being typed. This property is only available at run time.

TextAlign

Specifies how text is aligned inside the object. The available options are:

- **1 - fmTextAlignLeft**: aligns the text to the left of the object
- **2 - fmTextAlignCenter**: aligns the text to the center of the object
- **3 - fmTextAlignRight**: aligns the text to the right of the object

TextLength

Returns the number of characters typed in the object. This property is only available at run time.

Value

It is the text in the edition area.

WordWrap

Enables or disables line breaks in the text, in case the available area of the text overrides the limits determined by the object. For this property to work, the **Multiline** property must be True.

4.3.2.12.11 Spin Button

This section contains information about **events** and **properties** of the Spin Button object. This object does not have methods associated to it.

4.3.2.12.11.1 Events

This section contains information about the events of the Spin Button object.

SpinDown

SpinDown()

It occurs whenever users press the down arrow key. This event decreases the object's **Value** property.

SpinUp

SpinUp()

It occurs whenever users press the up arrow key. This event increases the object's **Value** property.

4.3.2.12.11.2 Properties

This section contains information about the properties of the Spin Button object.

Delay

Specifies the object's delay. The **Delay** property affects the time duration between consecutive **SpinUp**, **SpinDown**, and **Change** events, generated when users click the Spin Button and keep it pressed. The first event occurs immediately. The waiting time until the second occurrence of the event is five times the value specified for this property. After this initial occurrence, the interval between events is the one specified at the **Delay** property. Default value is 50 ms. This means that the object starts the first event after 250 ms (five times the specified value), and start the following events after every 50 ms.

Max

Determines the object's maximum limit.

Min

Determines the object's minimum limit.

Orientation

Determines object's orientation on Screen. The options available for this property are:

- **-1 - fmOrientationAuto**: determines orientation automatically, based on the object's dimensions, that is, how it was created
- **0 - fmOrientationVertical**: the object is placed vertically
- **1 - fmOrientationHorizontal**: the object is placed horizontally

Default value is **-1 - fmOrientationAuto**.

SmallChange

Specifies the amount of movement that happens when users click one of the object's scroll arrows. Default value is 1.

Value

Integer between the values defined for **Min** and **Max** properties. It indicates spin's initial value. It does not accept values lower than **Min** or higher than **Max**.

4.3.2.12.12 Scrollbar

This section contains information about **events** and **properties** of the Scrollbar object. This object does not have methods associated to it.

4.3.2.12.12.1 Events

This section contains information about the events of the Scrollbar object.

Scroll

Scroll()

The **Scroll** event is generated when the scroll bar pointer is moved to any direction.

4.3.2.12.12.2 Properties

This section contains information about the properties of the Scrollbar object.

Delay

Specifies the object's delay. The **Delay** property affects the time duration between consecutive **SpinUp**, **SpinDown**, and **Change** events, generated when users click the Scrollbar and keep it pressed. The first event occurs immediately. The waiting time until the second occurrence of the event is five times the value specified for this property. After this initial occurrence, the interval between events is the one specified at **Delay** property. Default value is 50 ms. This means that the object starts the first event after 250 ms (five times the specified value), and start the following events after every 50 ms.

LargeChange

Specifies the number of steps for the Scrollbar's cursor. The **LargeChange** property's value is the quantity that changes the **Value** property when users click the area between Scrollbar's box and the Scrollbar's cursor. Any integer is allowed for the **LargeChange** property, but the recommended interval is from -32,767 to 32,767. Also, this value should be between the values of **Max** and **Min** properties.

Max

Determines the object's maximum limit.

Min

Determines the object's minimum limit.

Orientation

Determines object's orientation on Screen. The available options for this property are:

- **-1 - fmOrientationAuto**: determines orientation automatically, based on the object's dimensions, that is, how it was created
- **0 - fmOrientationVertical**: the object is placed vertically
- **1 - fmOrientationHorizontal**: the object is placed horizontally

Default value is **-1 - fmOrientationAuto**.

ProportionalThumb

The **ProportionalThumb** property specifies whether the Scrollbar size is equal to the object's dimension. If this property is set to True, the Scrollbar box has the same dimensions of the object. Otherwise, if the object is resized, the Scrollbar box remains with its original size. The default value of this property is True.

SmallChange

Specifies the amount of movement that happens when users click one of the object's scroll arrows. Default value is 1.

Value

Integer between the values defined for **Min** and **Max** properties. It indicates Scrollbar's initial value. It does not accept values lower than **Min** or higher than **Max**.

4.3.2.13 E2Controls

This section contains information about events, methods, and properties of E2Control objects.

4.3.2.13.1 Common Properties

This section contains information about the common properties of the E2Control objects.

4.3.2.13.1.1 Frame_BorderColor

Defines a color for the object's frame.

4.3.2.13.1.2 Frame_BorderEnabled

Enables or disables the object's frame.

4.3.2.13.1.3 Frame_BorderThickness

Defines the thickness of the object's frame, in pixels.

4.3.2.13.1.4 Frame_Color

Defines the background color of the object's title area. The default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.13.1.5 Frame_Enable

◩ Enables or disables the display of the object's frame. The default value of this property is True, except for the **E2Button** object.

4.3.2.13.1.6 Frame_Enable3D

◩ Enables or disables a 3D effect for the object's frame.

4.3.2.13.1.7 Frame_Separator

◩ Enables or disables the display of a separator line between the title and the object.

4.3.2.13.1.8 Frame_Set3DInset

◩ If this property is set to True, the object's border appears as lowered. If it is False (default), the object's border appears as raised.

4.3.2.13.1.9 Frame_Thickness3D

9 Defines the thickness of the 3D border of the object's frame, in pixels.

4.3.2.13.1.10 Frame_Title

A This property defines the title of the object's frame.

4.3.2.13.1.11 Frame_TitleColor

Defines the font color of the frame's title. The default value of this property is black (**RGB(0, 0, 0)**).

4.3.2.13.1.12 Frame_TitleEnabled

◩ Enables or disables the display of the frame's title. The default value of this property is True.

4.3.2.13.1.13 Frame_TitleFont

A The **Frame_TitleFont** property is used to determine the font of the frame's title. This property cannot be used in scripts or Links, being set only via Studio.

4.3.2.13.2 E2Animation

This section contains information about **properties** of the E2Animation object. This object does not have events nor methods associated to it.

4.3.2.13.2.1 Properties

This section contains information about the properties of the E2Animation object.

BackgroundColor

This property defines the background color of an E2Animation object. The default value of this property is white (**RGB(255, 255, 255)**).

BlinkTime

Defines the time interval, in milliseconds, of the object's blinking effect.

Border

Enables or disables the display of a border around the object.

DefaultZone

Defines the default Zone, which is displayed when the associated Tag is out of the limits of other Zones defined for this object.

IsTransparent

If the value of this property is True, defines that the object's background is transparent, allowing the screen background to appear. Otherwise, the background color is solid, as defined in the **BackgroundColor** property.

Value

This property defines a value that determines what is the active Zone. The default value of this property is 0.

Zones

Collection of Zones of an E2Animation object.

4.3.2.13.2.2 Zone Collection

This section contains information about **methods** of the Zone Collection object. This object does not have events nor methods associated to it.

Methods

This section contains information about the methods of the Zone Collection object.

Add

Add([*AxisName*])

Adds a new Zone into the Zone Collection. The *AxisName* parameter is optional, being kept for compatibility reasons.

Remove

Remove(*Index*)

Removes a Zone. The *Index* parameter indicates the index of the Zone to be removed.

4.3.2.13.2.3 Zones

Defines a set of bitmap images used to create an animation effect in the object. The Zones can be configured by accessing the object's Properties window, on the **E2Animation** tab. The options for this tab are the following:

Available options for the E2Animation tab

OPTION	DESCRIPTION
Zones	List with all Zones defined in the object.
Add button	Adds a new Zone.
Remove button	Erases the selected Zone.
Default zone	Defines the selected Zone as the object's default Zone.
Blink	Defines if the bitmap blinks when the object's value is inside the Zone's interval.
Tip	Displays a help text over the Zone.
Minimum	Minimum value for the Zone.
Maximum	Maximum value for the Zone.
Image file	Name of the bitmap file displayed when the object's value is inside the Zone's interval.
Example	Displays a preview of the bitmap file of the selected Zone.

Properties

This section contains information about the properties of the Zones of the E2Animation object.

Blink

◩ Indicates that this Zone is part of the blinking effect. The default value of this property is False.

Filename

A Indicates what is the image filename used in this Zone.

Maximum

9 Defines the maximum value for this Zone. The default value of this property is 20000.

Minimum

9 Defines the minimum value for this Zone. The default value of this property is 0.

TipEnable

◩ Enables or disables a tip for this Zone. The default value of this property is False.

TipText

A Defines a tip for this Zone. The default value of this property is an empty **String**.

4.3.2.13.3 E2Bitmap

This section contains information about **properties** of the E2Bitmap object. This object does not have events nor methods associated to it.

4.3.2.13.3.1 Properties

This section contains information about the properties of the E2Bitmap object.

Filename

Defines the image file name linked to this E2Bitmap. The file path may be the full file path on disk, as well as a relative application path (when the image file is inserted as an application Resource). The default value of this property is an empty **String**.

IsTransparent

◩ This property enables or disables object transparency, based on the color defined by the **TransparentColor** property.

TransparentColor

Defines what is the color to be considered by the **IsTransparent** property as transparent. The default value of this property is white (**RBG(255, 255, 255)**).

4.3.2.13.4 E2Button

This section contains information about **events** and **properties** of the E2Button object. This object does not have methods associated to it.

4.3.2.13.4.1 Events

This section contains information about the events of the E2Button object.

OnRelease

OnRelease()

This event is triggered when the mouse button is released.

4.3.2.13.4.2 Properties

This section contains information about the properties of the E2Button object.

Action

This property defines the behavior of an E2Button object when clicked. The available values for this property are:

- **0 - Momentary**: Standard button behavior, appearing lowered only when pressing the mouse
- **1 - Toggle**: It has two states, on and off
- **2 - Jog**: Switches between two values, one when the button is pressed and another one when the button is released

The default value of this property is **0 - Momentary**.

Alignment

Determines the text alignment of an E2Button. The available values for this property are:

- **0 - HorizontalAlignmentLeft**: Aligns the text to the left
- **1 - HorizontalAlignmentCenter**: Aligns the text to the center
- **2 - HorizontalAlignmentRight**: Aligns the text to the right

The default value of this property is **1 - HorizontalAlignmentCenter**.

BackgroundColor0

Defines the E2Button's background color when it is not pressed. The default value of this property is gray (**RGB(192, 192, 192)**).

BackgroundColor1

Defines the E2Button's background color when it is pressed. The default value of this property is gray (**RGB(192, 192, 192)**).

Bitmap0

Defines the E2Button's image when it is not pressed. The default value of this property is an empty **String**.

Bitmap1

Defines the E2Button's image when it is pressed. The default value of this property is an empty **String**.

Text0

Defines the E2Button's text when it is not pressed. The default value of this property is "OFF".

Text1

Defines the E2Button's text when it is pressed. The default value of this property is "ON".

TextColor0

Defines the E2Button's text color when it is not pressed. The default value of this property is black (**RGB(0, 0, 0)**).

TextColor1

Defines the E2Button's text color when it is not pressed. The default value of this property is black (**RGB(0, 0, 0)**).

TextFont0

This property is used to determine the E2Button's font when it is not pressed. This property cannot be used in scripts or Links, being set only via Studio.

TextFont1

This property is used to determine the E2Button's font when it is pressed. This property cannot be used in scripts or Links, being set only via Studio.

Type

Defines the E2Button's type. The available values for this property are the following:

- **0 - ButtonTypeKey**: standard button behavior
- **1 - ButtonTypeSwitchH**: the button behavior is a horizontally divided switch
- **2 - ButtonTypeSwitchV**: the button behavior is a vertically divided switch
- **3 - ButtonTypeLeverH**: the button behavior is a lever that moves from left to right and vice versa
- **4 - ButtonTypeLeverV**: the button behavior is a lever that moves from top to bottom and vice versa
- **5 - ButtonTypeTransparent**: the button is transparent
- **6 - ButtonTypeUserBitmap**: the button display switches between the images defined in the **Bitmap0** and **Bitmap1** properties
- **7 - ButtonTypeCheckbox**: the button behavior is the same as a Check Box
- **8 - ButtonTypeRadio**: the button behavior is the same as a Radio Box

The default value of this property is **0 - ButtonTypeKey**.

Value

The **Value** property is a **Variant** that assumes the value in the **Value0** property if the button is not pressed, and the value in the **Value1** property if the button is pressed.

Value0

Defines the value of the **Value** property when the button is not pressed.

Value1

Defines the value of the **Value** property when the button is pressed.

4.3.2.13.5 E2Display

This section contains information about **properties** of the E2Display object. This object does not have events nor methods associated to it.

4.3.2.13.5.1 Properties

This section contains information about the properties of the E2Display object.

BackgroundColor

This property defines the object's background color. The default value of this property is gray (**RGB(192, 192, 192)**).

BackgroundStyle

Defines the object's background style. The available values of this property are the following:

- **0 - bsTransparent**: background is transparent
- **1 - bsOpaque**: the color defined in the **BackgroundColor** property is visible

The default value of this property is **1 - bsOpaque**.

Format

Contains a text that represents a mask where object values are displayed. This mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for

displaying negative numbers as signed or between parentheses. For very large or very small numbers, the **Scientific** format is recommended

- **Date**: Displays numerical date values (when valid). To represent only time, use the equivalent format
- **Time**: Displays numerical time values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies the number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays the number with a mantissa and exponent notation. Ideal for numbers with variable magnitude. Allows up to 15 decimal places
- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal, or binary, for example)
- **Other**: Allows directly editing the format code, or selecting a previously created format

HorizontalAlignment

Defines the horizontal alignment of an E2Display's text. The available values of this property are the following:

- **0 - HorizontalAlignmentLeft**: aligns horizontally to the left
- **1 - HorizontalAlignmentCenter**: aligns horizontally to the center
- **2 - HorizontalAlignmentRight**: aligns horizontally to the right

The default value of this property is **1 - HorizontalAlignmentCenter**.

MultiLine

Defines whether the object has multiple lines or not. This property is only available if the **Value** property is a **String**-type.

TextColor

Defines the object's text color. The default value of this property is black (**RGB(0, 0, 0)**).

TextFont

The **TextFont** property is used to determine the object's font. This property cannot be used in Links. See the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

Value

This property contains a **Variant** that can assume any data type value, and the

way these values are displayed is defined in the **Format** property.

VerticalAlignment

Defines the vertical alignment of an E2Display's text. The available values for this property are the following:

- **0 - VerticalAlignmentTop**: aligns vertically with the upper side of the object
- **1 - VerticalAlignmentMiddle**: aligns vertically with the center of the object
- **2 - VerticalAlignmentBottom**: aligns vertically with the bottom side of the object

The default value of this property is **1 - VerticalAlignmentMiddle**.

4.3.2.13.6 E2Gauge

This section contains information about **properties** of the E2Gauge object. This object does not have events nor methods associated to it.

4.3.2.13.6.1 Properties

This section contains information about the properties of the E2Gauge object.

BackgroundColor

This property defines the object's background color. The default value of this property is gray (**RGB(128, 128, 128)**).

BulletsVisible

Displays or hides the **bullet**-type scale marks.

DecimalPlaces

9 This property defines the number of decimal places for the E2Gauge's nominal value.

FrameColor

Defines the object's background color.

HiColorLegend

Defines the High limit's legend color. The default value of this property is yellow (**RGB(255, 255, 0)**).

HiDiv

9 Sets the beginning of the scale for the High limit. The default value of this property is 13300.

HiHiColorLegend

Defines the Very High limit's legend color. The default value of this property is red (**RGB(255, 0, 0)**).

HiHiDiv

Sets the beginning of the scale for the Very High limit. The default value of this property is 16600.

HiHiLimitVisible

Enables or disables the display of the Very High limit.

HiLimit

The maximum value of this property is 1, and the minimum is limited by the **LowLimit** property. The default value of this property is 0,7.

HiLimitVisible

Enables or disables the display of the High limit.

LegendVisible

Displays a bar along the E2Gauge object, which allows configuring different colors, depending on a range of values. The default value of this property is True.

LimitVisible

Defines whether the scale's minimum and maximum values appear on the chart or not.

LowColorLegend

Defines the Low limit's legend color. The default value of this property is dark green (**RGB(0, 128, 0)**).

LowDiv

Sets the beginning of the scale for the Low limit. The default value of this property is 6600.

LowLimit

The minimum value for this property is 0,1, and the maximum is limited by the **HiLimit** property. The default value of this property is 0,62.

LowLimitVisible

Enables or disables the display of the Low limit.

LowLowColorLegend

Defines the Very Low limit's legend color. The default value of this property is

green (**RGB(0, 255, 0)**).

LowLowDiv

9 Sets the beginning of the scale for the Very Low limit. The default value of this property is 3300.

LowLowLimitVisible

◩ Enables or disables the display of the Very Low limit.

Maximum

9 Defines the maximum value of an E2Gauge's scale.

Minimum

9 Defines the minimum value of an E2Gauge's scale.

NeedleColor

Defines the E2Gauge's needle color. The default value of this property is white (**RGB(255, 255, 255)**).

NeedleThickness

9 Defines the E2Gauge needle's thickness, in pixels. The default value of this property is 2, and it only accepts 1 or 2 as its value.

NormalColor

Defines the Normal limit's legend color. The default value of this property is olive (**RGB(128, 128, 0)**).

NumberOfPoints

9 Defines the number of visible subdivisions on the object's scale.

Orientation

Defines the E2Gauge's orientation. The available values for this property are the following:

- **0 - Left**: the lower part of the object is aligned with the left side of the frame
- **1 - Up**: the lower part of the object is aligned with the upper part of the frame
- **2 - Down**: the lower part of the object is aligned with the lower part of the frame
- **3 - Right**: the lower part of the object is aligned with the right side of the frame

The default value of this property is **2 - Down**.

Reverted

◩ Enables or disables reversing the object's scale.

ShowFrame

◩ Enables or disables displaying the background along the needle's path.

StartAngle

9 Defines the initial displaying angle of the E2Gauge's needle.

SubTickColor

Defines the scale's subdivision color. The default value of this property is black (**RGB(0, 0, 0)**).

SubTicksVisible

◩ Enables or disables displaying the scale's subdivisions.

TextColor

Defines the scale's text color. The default value of this property is black (**RGB(0, 0, 0)**).

TextFont

A The **TextFont** property is used to determine the scale's text font. This property cannot be used in Links. See the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

ThickTicks

◩ Enables or disables displaying thicker scale dividers. The default value of this property is False.

TickColor

Defines the scale's divider colors. The default value of this property is black (**RGB(0, 0, 0)**).

TicksVisible

◩ Enables or disables displaying scale dividers.

TickValues

◩ Enables or disables displaying scale divider values.

TotalNumberOfSubTicks

9 Defines the total amount of subdivisions displayed on the scale.

Value

This property defines a value between the **Maximum** and **Minimum** properties of the object's scale.

ValueVisible

Enables or disables displaying the value of the **Value** property. The default value of this property is False.

4.3.2.13.7 E2Setpoint

This section contains information about **properties** of the E2Setpoint object. This object does not have events nor methods associated to it.

4.3.2.13.7.1 Properties

This section contains information about the properties of the E2Setpoint object.

AutoSend

If this property is True (default), the value defined in the **Value** property is updated on Links, as soon as the object loses focus. Otherwise, Links only receive this value when the ENTER key is pressed.

BackgroundColor

This property defines the object's background color. The default value of this property is gray (**RGB(192, 192, 192)**).

BackgroundStyle

Defines the object's background style. The values of this property are the following:

- **0 - bsTransparent**: background is transparent
- **1 - bsOpaque**: color defined in the **BackgroundColor** property is visible

The default value of this property is **1 - bsOpaque**.

EnableMaxLimit

Enables or disables the definition of a maximum limit for E2Setpoint's value.

EnableMinLimit

Enables or disables the definition of a minimum limit for E2Setpoint's value.

Format

Contains a text that represents a mask where object values are displayed. This

mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for displaying negative numbers as signed or between parentheses. For very large or very small numbers, the **Scientific** format is recommended
- **Date**: Displays numerical date values (when valid). To represent only time, use the equivalent format
- **Time**: Displays numerical time values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies the number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays the number with a mantissa and exponent notation. Ideal for numbers with variable magnitude. Allows up to 15 decimal places
- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal, or binary, for example)
- **Other**: Allows directly editing the format code, or selecting a previously created format

HorizontalAlignment

Defines the horizontal alignment of the E2Setpoint's text. The available values for this property are the following:

- **0 - HorizontalAlignmentLeft**: aligns horizontally to the left
- **1 - HorizontalAlignmentCenter**: aligns horizontally to the center
- **2 - HorizontalAlignmentRight**: aligns horizontally to the right

The default value of this property is **1 - HorizontalAlignmentCenter**.

HScroll

Enables or disables displaying a horizontal scroll bar on the text, in case the **MultiLine** property is True.

MaxLimit

9 Maximum limit that can be reached by the object's **Value** property. The default value of this property is 200. This limit is only checked if the **EnableMaxLimit** property is enabled.

MinLimit

Minimum limit that can be reached by the object's **Value** property. The default value of this property is 0. This limit is only checked if the **EnableMinLimit** property is enabled.

MultiLine

Defines whether the object has multiple lines or not. This property is only available if the **Value** property is of **String**-type.

ReadOnly

Indicates whether this object can be edited or not at run time. The default value of this property is False.

Refresh

Indicates whether the E2Setpoint's value is updated or not, as soon as Tag value changes. The default value of this property is True.

SelectAllOnFocus

Enables or disables selecting all characters of an E2Setpoint when this object receives the focus. The default value of this property is True.

TextColor

Defines the object's text color. The default value of this property is black (**RGB(0, 0, 0)**).

TextFont

The **TextFont** property is used to determine the object's font. This property cannot be used in Links. See the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

Type

Defines the Setpoint type. The available values for this property are the following:

- **0 - setpointString**: accepts any alphanumerical characters
- **1 - setpointNumeric**: accepts only numerical characters and the decimal separator (dot or comma, according to regional settings)
- **2 - setpointDateTime**: accepts only date and time values, which are converted to the format defined on regional settings)

The default value of this property is **1 - setpointNumeric**.

Value

This property defines a value for an E2Setpoint. The way this value is viewed is defined in the **Format** property.

VerticalAlignment

Defines the vertical alignment of the E2Setpoint's text. The available values of this property are the following:

- **0 - VerticalAlignmentTop**: aligns vertically to the top of the object
- **1 - VerticalAlignmentMiddle**: aligns vertically to the center of the object
- **2 - VerticalAlignmentBottom**: aligns vertically to the bottom of the object

The default value of this property is **1 - VerticalAlignmentMiddle**.

VScroll

Enables or disables displaying a vertical scroll bar on the text, in case the **MultiLine** property is True.

4.3.2.13.8 E2Text

This section contains information about **properties** of the E2Text object. This object does not have events nor methods associated to it.

4.3.2.13.8.1 Properties

This section contains information about the properties of the E2Text object.

BlinkTime

Defines the time interval, in milliseconds, of the object's blinking effect.

DefaultZone

Defines the object's default Zone.

Value

This property contains a **Variant** that may assume any data type values (**Integer**, **Boolean**, **String**, etc.).

Zones

Collection of Zones of an E2Text object.

4.3.2.13.8.2 Zone Collection

This section contains information about the **methods** of the Zone Collection object of the E2Text. This object does not have events nor properties associated to it.

Methods

This section contains information about the methods of the Zone Collection object of the E2Text.

Add

Add([*AxisName*])

Adds a new Zone in the Zone Collection. The *AxisName* parameter is optional and has no effect, being kept for compatibility reasons.

Remove

Remove(*Index*)

Removes a Zone. The *Index* parameter indicates the index of the Zone to be removed.

4.3.2.13.8.3 Zones

Defines a set of Zones for an E2Text object. These Zones can be configured by accessing the object's Properties window, on the **Zones** tab. The available options on this tab are the following:

Available options on the Zones tab

OPTION	DESCRIPTION
Zones	List with all Zones defined for this object.
Add button	Adds a new Zone.
Remove button	Removes the selected Zone.
Blinks every (ms)	Defines whether the Zone's text and background blink when the object's value is inside the Zone's interval.
Message	Text of the message displayed when the object's value is inside the Zone's interval.
Alignment	Defines text alignment.
Font	Defines text font.
Background Color	Defines Zone's background color.
Transparent	Defines if the object's background is transparent when the Zone is active.
Default zone	Defines the selected Zone as the object's default Zone.
Blink	Defines if the Zone blinks when the object's value is inside the Zone's interval.

OPTION	DESCRIPTION
Minimum	Minimum value for the Zone.
Maximum	Maximum value for the Zone.
Tip	Displays a help text for the Zone.
Example	Displays an example of the Zone's behavior at run time.

Properties

This section contains information about the properties of the E2Text object Zones.

BackgroundColor

Defines a text Zone's background color. The default value of this property is white (**RGB(255, 255, 255)**).

Blink

Indicates that this Zone participates in the blinking effect. The default value of this property is False. If it is enabled, this zone switches with the default Zone, according to the time defined in the E2Text's **BlinkTime** property.

HorizontalAlignment

Defines the text alignment. The available values are:

- **0 - HorizontalAlignmentLeft**: aligns horizontally to the left
- **1 - HorizontalAlignmentCenter**: aligns horizontally to the center
- **2 - HorizontalAlignmentRight**: aligns horizontally to the right

The default value of this property is **1 - HorizontalAlignmentCenter**.

Maximum

Defines the Zone's maximum value. The default value of this property is 20000.

Message

Defines the text linked to the Zone. This message is displayed when the associated Tag is inside the Zone limits.

Minimum

Defines the Zone's minimum value. The default value of this property is 0.

TextColor

Defines the Zone's text color. The default value of this property is black (**RGB(0, 0, 0)**).

TextFont

Defines style, color, and size of the font used to display the message text. This property cannot be used in Links. See the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

TipEnable

Enables or disables a tip for the Zone. The default value of this property is False.

TipText

Defines a tip for the Zone. The default value of this property is an empty **String**.

Transparent

Defines that the object background is transparent when this Zone is active.

4.3.2.14 Elipse KeyPad

This section contains information about **methods** and **properties** of the Elipse KeyPad object. This object does not have events associated to it.

4.3.2.14.1 Methods

This section contains information about the methods of the Elipse KeyPad object.

4.3.2.14.1.1 Hide

Hide()

Hides the Elipse KeyPad. This method has no effect if the KeyPad is already invisible. Example:

```
' The following code hides the KeyPad
Sub CommandButton1_Click()
  Application.GetKeyPad().Hide()
End Sub
```

4.3.2.14.1.2 Show

Show()

Displays the Elipse KeyPad. This method has no effect if the KeyPad is already visible. Example:

```
' The following code displays the KeyPad
Sub CommandButton1_Click()
  Application.GetKeyPad().Show()
End Sub
```

4.3.2.14.2 Properties

This section contains information about the properties of the Elipse KeyPad object.

4.3.2.14.2.1 AutoHideOnEnter

◩ Automatically hides the KeyPad when the virtual keyboard's ENTER key is pressed.
Example:

```
' Changes the Keypad behavior.
' If the ENTER key of the virtual keyboard is pressed
' the KeyPad is hidden.
Sub CommandButton1_Click()
  Application.GetKeyPad().AutoHideOnEnter = True
End Sub
```

4.3.2.14.2.2 AutoHideOnEsc

◩ Automatically hides the KeyPad when the virtual keyboard's ESC key is pressed.
Example:

```
' Changes the KeyPad behavior.
' If the ESC key of the virtual keyboard is pressed
' the KeyPad is hidden.
Sub CommandButton1_Click()
  Application.GetKeyPad().AutoHideOnEsc = True
End Sub
```

4.3.2.14.2.3 Layout

A Allows changing the KeyPad's presentation layout. Possible values of this property are:

- **br-simple**: Displays an alphanumeric keyboard

Example of an alphanumeric KeyPad

- **num**: Displays a numeric keyboard

Example of a numeric KeyPad

Example:

```
' Switches between Alphanumeric and Numeric modes
Sub ToggleButton1_Click()
  If ToggleButton1.Value Then
    Application.GetKeyPad().Layout = "br-simple"
  Else
    Application.GetKeyPad().Layout = "num"
  End If
End Sub
```

4.3.2.14.2.4 SizeFactor

9 Increases or decreases the KeyPad's original size, by using a multiplication factor. The KeyPad's original size is the following:

Default size values for the KeyPad

LAYOUT	WIDTH	HEIGHT
Alphanumeric	550 px	250 px
Numeric	200 px	300 px

The following example resizes the KeyPad to 75% of its original size.

```
Sub CommandButton1_Click()
  Application.GetKeyPad().SizeFactor = 0.75
```

```
  Application.GetKeyPad().Show()
End Sub
```

4.3.2.14.2.5 Sound

A Allows changing the sound played when clicking a KeyPad key. The default value of this property is an empty **String**, meaning that no sound plays when clicking a key. In case the value of this property changes, the new value must be an absolute path of a WAV file, or this file must be added to the Domain as a Resource. Example:

```
Sub CommandButton1_Click()
  Application.GetKeyPad().Sound = "c:\windows\media\ringout.wav"
End Sub
```

4.3.2.14.2.6 X

9 The X coordinate of the KeyPad's upper left corner. This property can be used to move the KeyPad horizontally. Example:

```
Sub CommandButton1_Click()
  ' Moves the KeyPad 50 pixels to the right
  Application.GetKeyPad().X = Application.GetKeyPad().X + 50
End Sub
```

4.3.2.14.2.7 Y

9 The Y coordinate of the KeyPad's upper left corner. This property can be used to move the KeyPad vertically. Example:

```
Sub CommandButton1_Click()
  ' Moves the KeyPad 50 pixels down
  Application.GetKeyPad().Y = Application.GetKeyPad().Y + 50
End Sub
```

4.4 E3Alarm

This section contains information about **events**, **methods**, and **properties** of the E3Alarm object.

4.4.1 Events

This section contains information about the events of the E3Alarm object.

4.4.1.1 KeyPress

KeyPress(*KeyAscii*)

This event occurs when the object has the keyboard focus, and the user presses a key corresponding to a character that can be showed on Screen (an ANSI key, with a code indicated by the *KeyAscii* parameter). That is, the event occurs when some of the following keys are pressed:

- Any keyboard character that can be printed
- The CTRL key combined with any standard alphabet character
- The CTRL key combined with any special character
- The BACKSPACE key
- The ESC key

This event does not occur in the following conditions:

- By pressing the TAB key
- By pressing the ENTER key
- By pressing the DEL key (this is not an ANSI key)
- By pressing keyboard arrow keys
- When a key change the focus from one object to another

While a user presses a key that produces an ANSI code, the object repeatedly receives the **KeyDown** and the **KeyPress** events. When the user releases the key, the **KeyUp** event occurs.

To monitor keyboard physical status, or handle keys not recognized by the **KeyPress** event (such as function keys, browsing keys, etc.), use the **KeyDown** and **KeyUp** events.

4.4.1.2 MouseMove

MouseMove()

Occurs when the mouse pointer is moved over the E3Alarm.

4.4.2 Methods

This section contains information about the methods of the E3Alarm object.

4.4.2.1 AboutBox

AboutBox()

This method displays a dialog box with information about E3Alarm's version and copyright.

4.4.2.2 AckAll

AckAll([*Operator*])

Allows acknowledging all alarms globally. The *Operator* parameter is an optional **String** indicating the name of the operator that acknowledged the alarm. This value is displayed on the E3Alarm's **Operator** column. If it is omitted, the current Viewer's user is used, or else "(No User)", in case there is no user logged in. For the acknowledgment itself, the logged in user must have permission to acknowledge alarms.

4.4.2.3 AckCurrentFilter

AckCurrentFilter([*Operator*])

Allows acknowledging all alarms of the current filter. The *Operator* parameter is an optional **String** indicating the name of the operator that acknowledged the alarm. This value is displayed on the E3Alarm's **Operator** column. If it is omitted, the current Viewer's user is used, or else "(No User)", in case there is no user logged in. For the acknowledgment itself, the logged in user must have permission to acknowledge alarms.

4.4.2.4 AckSelected

AckSelected([*Operator*])

Allows acknowledging the selected alarms. If there is no selected alarm in the E3Alarm, this method fails. The user may acknowledge the alarm (in this case, a new record is inserted into the Database, indicating the acknowledgment), and in the E3Alarm the corresponding row indicates that it was acknowledged. The *Operator* parameter is an optional **String** indicating the name of the operator that acknowledged the alarm. This value is displayed on the E3Alarm's **Operator** column. If it is omitted, the current Viewer's user is used, or else "(No User)", in case there is no user logged in. For the acknowledgment itself, the logged in user must have permission to acknowledge alarms.

4.4.2.5 GetFocusedEvent

GetFocusedEvent()

This method returns an object with the properties of the selected event (the one that has focus) on E3Alarm, in case there is a selected event. If no event is selected, this method returns `Nothing`.

The properties of the returned object contains field values of the selected event. This object contains a copy of the values at the time of the method call, therefore if the selected event changes, these properties are not updated automatically, being necessary to use this method every time users need to get updated information

about the selected event. The object properties returned by this method are the following:

Properties of the object returned by the GetFocusedEvent method

NAME	DESCRIPTION
Acked	Informs whether the alarm was acknowledged or not. This field can assume values **0**: not acknowledged, and **1**: acknowledged.
AckRequired	Determines the automatic acknowledgment of the alarm. This field has the values **0**: automatic acknowledgment, and **1**: manual acknowledgment.
AckTime	Records E3's date and time at the moment the alarm is acknowledged, or zero (12/30/1899) if the alarm has not been acknowledged yet. In case of alarms that do not need acknowledgment, this field assumes E3's date and time at the moment the alarm became active.
ActorID	Name of the operator that acknowledged the alarm. It can be: • The Viewer's logged in user, when the acknowledgment is performed on E3Alarm (or "No User", if there is no logged in user). • "System", when the acknowledgment is automatic (that is, for alarms that ask for acknowledgment). • A name passed by Script (for example, using the Alarm Server's **AckArea**, **AckAllAlarms**, **LogTrackingEvent** methods, or the Alarm Source's **Ack** method). The limit of this field, when stored on a database, is 50 characters.
AlarmSourceName	Records the name of the Alarm Source. The limit of this field, when stored on a database, is 100 characters. For example, "AlarmSource1".
Area	For alarm events, the name of the Area to which the Alarm Source belongs. For other events (for example, using the Alarm Server's **LogTrackingEvent** method), may be a user-defined text. The limit of this field, when stored on a database, is 100 characters.
ConditionActive	Indicates whether the Alarm Source is in alarm. This field presents the states **0**: inactive condition and **1**: active condition.

NAME	DESCRIPTION
ConditionName	Name of the condition, if this is an alarm event. This field can have the following values: • **Dead Band**: Dead band alarm source type • **Digital**: Digital alarm source type • **Level**: Analog alarm source type • **RateOfChange**: Rate of change alarm source type If the event is not an alarm (for example, by using the Alarm Server's **LogTrackingEvent** method), this value is always an empty **String**. The limit of this field, when stored on a database, is 100 characters.
CurrentValue	Determines the value of the Alarm Source (converted to **Double**) at the moment of the event. For other events (for example, using the Alarm Server's **LogTrackingEvent** method), this value is always 0. The limit of this field, when stored on a database, is 100 characters.
Enabled	Determines whether the alarm check is enabled. This field presents the states **0**: disabled alarm source check and **1**: enabled alarm source check.
EventCategory	Category of the event. For alarms, this field can assume the following values: • **Dead Band**: Dead band alarm source type • **Digital**: Digital alarm source type • **Level**: Analog alarm source type • **RateOfChange**: Rate Of Change alarm source type For other events (for example, using the Alarm Server's **LogTrackingEvent** method), this field can assume user-defined values. The limit of this field, when stored on a database, is 100 characters.
EventCLSID	Unique identifier of an alarm. When a new alarm occurs on a Source, a new EventCLSID number is generated, so it keeps the same CLSID on the database, as long as it remains on the active and non acknowledged list of alarms.
EventTime	Date and time of the Alarm Source's value at the moment of the event.

NAME	DESCRIPTION
EventTimeUTC	Date and time of the Alarm Source's value at the moment of the event, relative to Greenwich Meantime. This value is the same as **EventTime**, being kept in E3 for compatibility reasons.
EventType	Type of the event. For alarm events, it is always "Condition". For other events, it can be a user-defined text, for example, using the Alarm Server's **LogTrackingEvent** method (for example, "Tracking", "Simple", etc.). The limit of this field, when stored on a database, is 100 characters.
FormattedValue	Shows the formatted value of the Alarm Source that is saved on the event. The limit of this field, when stored on a database, is 100 characters. **NOTE**: this is a read-only field.
FullAlarmSourceName	Records the full path of the Alarm Source, including areas, alarm configuration name, and possible folders where this Alarm Source might be inserted. For example, "Folder1.AlarmConfig1.Area1.AlarmSource1".
InTime	Records the date and time of the value when it gets into an alarm condition.
Message	It is the configured text on the Alarm Source, or specified by another event (for example, by using the Alarm Server's **LogTrackingEvent** method). The limit of this field, when stored on a database, is 200 characters.
OutTime	Records the date and time of the value when it gets out of an alarm condition, or zero (12/30/1899) if the alarm does not get out of the active condition.
Quality	Quality of the Alarm Source's value at the moment of the event. This field assumes these numerical values: • **0 - 63**: bad quality • **64 - 127**: uncertain quality • **128 - 191**: undefined value • **192 - 255**: good quality If the event is not an alarm (for example, if it is using the Alarm Server's **LogTrackingEvent** method), this field is equal to an empty **String**. Example: "Bad (0)", "Uncertain (64)", "?? (128)", "Good (192)".

NAME	DESCRIPTION
Severity	It is the configured severity value on the Alarm Source. This field can assume values **0**: high; **1**: medium; or **2**: low. It can also assume another user-defined value if it is an event, as when it is using the Alarm Server's **LogTrackingEvent** method.
Source	For alarm events, informs the expression used to evaluate alarm conditions. The limit of this field, when stored on a database, is 100 characters. For example, "Data.DemoTag1.Value".
SubConditionName	Name of the sub-condition, if it is an alarm event. This field can assume the values: • **DB**: Dead band alarm • **DIG**: Digital alarm • **RC**: Rate of change alarm • **LOLO**: Very low range analog alarm • **LO**: Low range analog alarm • **HI**: High range analog alarm • **HIHI**: Very high range analog alarm If the event is not an alarm (for example, if it is using the Alarm Server's **LogTrackingEvent** method), this field is equal to an empty **String**. The limit of this field, when stored on a database, is 100 characters.
UserField	Read-only indexed property, whose index varies from 1 to the total amount of user-defined fields.

Example:

```
Dim evt
Set evt = GetFocusedEvent()
If Not(evt Is Nothing) Then
  Dim str
  str = str & "EventTime = " & evt.EventTime & Chr(13)
  str = str & "EventTimeUTC = " & evt.EventTimeUTC & Chr(13)
  str = str & "InTime = " & evt.InTime & Chr(13)
  str = str & "OutTime = " & evt.OutTime & Chr(13)
  str = str & "AckTime = " & evt.AckTime & Chr(13)
  str = str & "CurrentValue = " & evt.CurrentValue & Chr(13)
  str = str & "ActorID = " & evt.ActorID & Chr(13)
  str = str & "Area = " & evt.Area & Chr(13)
  str = str & "AlarmSourceName = " & evt.AlarmSourceName &
Chr(13)
  str = str & "FullAlarmSourceName = " & evt.FullAlarmSourceName
& Chr(13)
```

```
  str = str & "ConditionName = " & evt.ConditionName & Chr(13)
  str = str & "EventCategory = " & evt.EventCategory & Chr(13)
  str = str & "EventType = " & evt.EventType & Chr(13)
  str = str & "Message = " & evt.Message & Chr(13)
  str = str & "Quality = " & evt.Quality & Chr(13)
  str = str & "Source = " & evt.Source & Chr(13)
  str = str & "SubConditionName = " & evt.SubConditionName &
Chr(13)
  str = str & "FormattedValue = " & evt.FormattedValue & Chr(13)
  str = str & "UserField(1) = " & evt.UserField(1) & Chr(13)
  str = str & "UserField(2) = " & evt.UserField(2) & Chr(13)
  str = str & "UserField(3) = " & evt.UserField(3) & Chr(13)
  str = str & "UserField(4) = " & evt.UserField(4) & Chr(13)
  str = str & "Severity = " & evt.Severity & Chr(13)
  str = str & "Acked = " & evt.Acked & Chr(13)
  str = str & "AckRequired = " & evt.AckRequired & Chr(13)
  str = str & "ConditionActive = " & evt.ConditionActive &
Chr(13)
  str = str & "Enabled = " & evt.Enabled & Chr(13)
  str = str & "EventCLSID = " & evt.EventCLSID & Chr(13)
  MsgBox str
Else
  MsgBox "No selected event."
End If
```

4.4.3 Properties

This section contains information about the properties of the E3Alarm object.

NOTE: E3 uses the Himetric system for coordinate and width definitions. In this system, each logical unit is equal to one hundreth of a centimeter. That is, each 1000 units correspond to one centimeter. Thus, this is the standard adopted when describing E3 properties, when applicable.

4.4.3.1 ActiveAlarms

9 Determines the total amount of active alarms in the object. This is a read-only property.

4.4.3.2 ActiveHighAlarms

9 Indicates the number of high severity active alarms. This is a read-only property.

4.4.3.3 ActiveHighNACKAlarms

9 Indicates the number of high severity unacknowledged alarms. This is a read-only property.

4.4.3.4 ActiveLowAlarms

9 Indicates the number of low severity active alarms. This is a read-only property.

4.4.3.5 ActiveLowNACKAlarms

9 Indicates the number of low severity unacknowledged alarms. This is a read-only property.

4.4.3.6 ActiveMedAlarms

9 Indicates the number of medium severity active alarms. This is a read-only property.

4.4.3.7 ActiveMedNACKAlarms

9 Indicates the number of medium severity unacknowledged alarms. This is a read-only property.

4.4.3.8 ActiveNACKAlarms

9 Indicates the total amount of unacknowledged alarms in the object (active or not). This is a read-only property.

4.4.3.9 AllowAckAll

◩ Enables an E3Alarm's contextual menu option that allows acknowledging all alarms. The default value of this property is True.

4.4.3.10 AllowAckCurrentFilter

◩ Enables an E3Alarm's contextual menu option that allows acknowledging all alarms of the current filter. If there is no visible alarms, this property has no effect. The default value of this property is True.

4.4.3.11 AllowAckSelected

◩ Enables an E3Alarm's contextual menu option that allows acknowledging all selected alarms. If there is no selected alarms, this property has no effect. The default value of this property is True.

4.4.3.12 AllowColumnClick

◩ Enables or disables field selection and their sort direction by clicking E3Alarm's column headers at run time. If True, and the header is invisible (see the **ColumnHeader** property), when users click the column title, data is sorted using this

column as the key. When users click the same column title again, the sort order is reversed (from ascending to descending, and vice-versa). When users click it with the SHIFT key pressed, this field is used as a second key. As in the primary key, a second click with SHIFT pressed reverses the sort order of the secondary field.

4.4.3.13 AlarmServer

A Name of the Alarm Server in the application.

4.4.3.14 AreaFilter

A Controls visible alarm areas in the E3Alarm. If its value is not an empty **String**, the events whose area name start with the indicated text are displayed. For example, if AreaFilter is equal to "Ana", then only Alarm areas such as "Analog.Production" or "Analysis" are displayed, but not "Digital.Analysis" or "Digital.Production". When the **SimpleAreaFilter** property is equal to False, the Alarm Area also allows using wildcards (such as * or ?) for filtering, and allowing multiple Area filters, separated by colons. The allowed wildcards are:

- **"*"**: Accepts none or any amount of characters
- **"?"**: Accepts any character
- **"#"**: Accepts any digit
- **"[ ]"**: Allows specifying a set of characters. For example:
  - **"[ab]"**: Accepts a character if it is "a" or "b"
  - **"[f-h]"**: Accepts a character between "f" and "h"
  - **"[!cz]"**: Accepts a character if it is neither "c" nor "z"
  - **"[!m-p]"**: Accepts any character, as long as it is not between "m" and "p"

Default value is an empty **String**, that is, there is no filter per Area (see also the **CustomFilter**, **ShowHighPriority**, **ShowMediumPriority**, and **ShowLowPriority** properties).

NOTE: The **AreaFilter** property corresponds to the **Filter** property, available until E3 version 4.0.

4.4.3.15 BannerMode

◩ Enables the visualization of a single message in the E3Alarm. The displayed message depends on the sort configuration and it is always selected.The default value of this property is False. For more information about the sort configuration, please check the topic **Sorting Tab**, in the **E3 User's Manual**.

4.4.3.16 BorderColor

Defines the E3Alarm's border color. The default value of this property is black (**RGB(0, 0, 0)**).

4.4.3.17 BorderThickness

Defines the E3Alarm's border thickness. The value of this property ranges from 0 (zero, which disables the border) to 10, and its default value is 1 (one).

4.4.3.18 ColumnHeader

When in True, this property enables viewing E3Alarm's header. The header also visually sorts data in the table (please check the **AllowColumnClick** property). Default value is True.

4.4.3.19 CustomFilter

Allows informing a customized filter for the alarms, as an expression. The following fields are available for usage on the filter expression:

- **AckRequired (Boolean)**: Indicates whether this message needs to be acknowledged.
- **Acked (Boolean)**: Indicates whether this message was already acknowledged.
- **AckTime (Date)**: Date and time when the alarm condition was acknowledged (or zero if it was not acknowledged).
- **ActiveSource (Integer)**: **-1**: None, **0**: ActiveSource, **1**: Scada, **2**: Operator, **3**: CCLink, **4**: Billing, **5**: Calculated, **6**: Database, **100**: TopologyProcessor, **101**: PowerFlow, **102**: StateEstimator, **103**: LoadShedding.
- **ActorID (String)**: Login of the user that acknowledged the message (or an empty **String** if the message was not yet acknowledged).
- **AlarmArea (String)**: Area of this alarm.
- **AlarmSourceName (String)**: Name of the Alarm Source object (only the name, not the full path).
- **ChangeMask (Integer)**: Field currently not used by E3, it is always 0 (zero).
- **ConditionName (String)**: Name of the last active alarm condition.
- **ConditionActive (Boolean)**: Indicates whether the alarm condition is active.
- **Cookie (Integer)**: Identifies an Alarm Source during an execution session.
- **CurrentValue (Double)**: Value of the Source at the moment the alarm condition became active.

- **Enabled (Boolean)**: Indicates whether the alarm verification on the Alarm Source is enabled.
- **EventCategory (String)**: Name of the alarm category (for example, "Level", "Rate of Change", "Dead Band", "Digital", or "Discrete").
- **EventTime (Date)**: Date and time of the last event update.
- **EventTimeUTC (Date)**: Date and time (UTC) of the last event update.
- **EventType (String)**: "Event" or "Condition" (alarm).
- **FullAlarmSourceName (String)**: Full name of the Alarm Source object.
- **FormattedValue (String)**: Contains the Source value (formatted) at the moment the alarm condition became active.
- **InTime (Date)**: Date and time the alarm condition became active.
- **Message (String)**: Alarm message.
- **OutTime (Date)**: Date and time the condition left the alarm (or zero if it is still active).
- **Quality (String)**: "Good (xxx)", "Bad (xxx)", or "Uncertain (xxx)".
- **Severity (Integer)**: **0**: High, **1**: Medium, or **2**: Low.
- **Source (String)**: Link of the Alarm Source.
- **SubConditionName (String)**: Name of the alarm sub-condition (for example, "LOLO", "LO", "HI", "HIHI", "DIG", etc.).

User-defined fields can also be used on the filter expression, by using the name defined on the Alarm Server.

Altogether, messages displayed on E3Alarm list always pass through five filters:

- Filter by type (alarm or event) (the **FilterType** property)
- Filter by severity (the **ShowLowPriority**, **ShowMediumPriority**, and **ShowHighPriority** properties)
- Filter by area (the **AreaFilter** and **SimpleAreaFilter** properties)
- Filter by the **CustomFilter** property
- Filter of the alarm summary (equivalent to the expression "Enabled AND (ConditionActive OR (AckRequired AND NOT Acked))")

Usage examples of the **CustomFilter** property:

- For a user-defined field called **IsSupressed**, and that users want to display only alarms where this field's value is different from zero, the expression to use is the

following:

```
IsSupressed <> 0
```

- To display only messages with the sub-condition "HIHI" or "LOLO" of an alarm object whose name starts with "Pressure", the expression to use is the following:

```
(SubConditionName = "HIHI" OR SubConditionName = "LOLO") AND
(Mid(AlarmSourceName, 1, 8) = "Pressure")
```

4.4.3.20 Domain

A Specifies the Domain to which E3Alarm connects. Default value is an empty **String**, that is, E3Alarm connects to the same Viewer's Domain where it is. For example, \\NameOfAnotherServer.

4.4.3.21 Enabled

◩ Enables the ActiveX object in the project. Default value is True.

4.4.3.22 FilterType

Performs the alarm filters. The available options are:

- **1 - OnlyAlarms**: displays only alarms
- **2 - OnlyEvents**: displays only events
- **3 - AlarmsAndEvents**: displays both alarms and events

4.4.3.23 Font

A Determines E3Alarm's header and row font. This is a read-only property, and cannot be modified at run time.

4.4.3.24 FourthSortAscending

◩ When set to False, the sort order of events by a fourth field is performed in descending order. Otherwise, it is performed in ascending order. The default value is False.

4.4.3.25 FourthSortField

A Determines the fourth field for sorting events in an E3Alarm. The field name must be always specified in English (check the available fields in the **E3 User's Manual**). The default value of this property is an empty **String**. This property has no effect when the **PrimarySortField**, **SecondarySortField**, or **ThirdSortField** properties are configured as an empty **String**.

4.4.3.26 GridBkColor

Determines E3Alarm's background color. Default value is the color configured in Windows for the **Window** item (**Control Panel - Display - Appearance - Advanced**).

4.4.3.27 InactiveHighNACKAlarms

Indicates the number of inactive and unacknowledged alarms with high severity. This is a read-only property.

4.4.3.28 InactiveLowNACKAlarms

Indicates the number of inactive and unacknowledged alarms with low severity. This is a read-only property.

4.4.3.29 InactiveMedNACKAlarms

Indicates the number of inactive and unacknowledged alarms with medium severity. This is a read-only property.

4.4.3.30 InactiveNACKAlarms

Determines the total amount of inactive and unacknowledged alarms. This is a read-only property.

4.4.3.31 PopupMenu

Enables a contextual menu on E3Alarm. Default value is True.

4.4.3.32 PrimarySortAscending

When False, the sort order of events by a primary field is performed in descending order. Otherwise, it is performed in ascending order. The default value is False.

4.4.3.33 PrimarySortField

Determines the primary field for sorting events in an E3Alarm. The field name must always be specified in English (check the available fields in the **E3 User's Manual**). The default value of this property is "EventTime". When this property is an empty **String**, the **SecondarySortField**, **ThirdSortField**, and **FourthSortField** properties do not have any effect.

4.4.3.34 SecondarySortAscending

When set to False, the sort order of events by a secondary field is performed in descending order. Otherwise, it is performed in ascending order. The default value

is False.

4.4.3.35 SecondarySortField

A Determines the secondary field for sorting events in an E3Alarm. The field name must always be specified in English (check the available fields in the **E3 User's Manual**). The default value of this property is an empty **String**. This property does not have any effect when the **PrimarySortField** property is an empty **String**.

4.4.3.36 ShowHighPriority

◩ Filters the alarms to be displayed, according to their severity. If True, high severity alarms are shown. Otherwise, these ones are hidden. Default value is True.

4.4.3.37 ShowLowPriority

◩ Filters the alarms to be displayed, according to their severity. If True, low severity alarms are shown. Otherwise, these ones are hidden. Default value is True.

4.4.3.38 ShowMediumPriority

◩ Filters the alarms to be displayed, according to their severity. If True, medium severity alarms are shown. Otherwise, these ones are hidden. Default value is True.

4.4.3.39 SimpleAreaFilter

◩ When in True, the behavior for filtering by Alarm Area name is based only on the coincidence of the initial part of the name. When in False, this behavior takes into account the whole Area name, but allows using wildcards and multiple area filters, which must be separated by colons. See also the **AreaFilter** property, which specifies filters per area name.

4.4.3.40 ThirdSortAscending

◩ When set to False, the sort order of events by a third field is performed in descending order. Otherwise, it is performed in ascending order. The default value is False.

4.4.3.41 ThirdSortField

A Determines the third field for sorting events in an E3Alarm. The field name must always be specified in English (check the available fields in the **E3 User's Manual**). The default value of this property is an empty **String**. This property has no effect when the **PrimarySortField** or **SecondarySortField** properties are configured as an empty **String**.

4.5 E3Browser

This section contains information about **events**, **methods**, and **properties** of the E3Browser object.

4.5.1 Events

This section contains information about the events of the E3Browser object.

4.5.1.1 KeyPress

KeyPress (*KeyAscii*)

This event occurs when the object has the keyboard focus, and the user presses a key corresponding to a character that can be showed on Screen (an ANSI key, with a code indicated by the *KeyAscii* parameter). That is, the event occurs when some of the following keys are pressed:

- Any keyboard character that can be printed
- The CTRL key combined with any standard alphabet character
- The CTRL key combined with any special character
- The BACKSPACE key
- The ESC key

This event does not occur in the following conditions:

- By pressing the TAB key
- By pressing the ENTER key
- By pressing the DEL key (this is not an ANSI key)
- By pressing keyboard arrow keys
- When a key change the focus from one object to another

While a user presses a key that produces an ANSI code, the object repeatedly receives the **KeyDown** and the **KeyPress** events. When the user releases the key, the **KeyUp** event occurs.

To monitor keyboard physical status, or handle keys not recognized by the **KeyPress** event (such as function keys, browsing keys, etc.), use the **KeyDown** and **KeyUp** events.

4.5.1.2 OnDrawRow

OnDrawRow(*Selected*, *Row*, *TextColor*, *BackColor*)

This method passes four parameters. The *Selected* parameter indicates whether the row is selected. The *Row* parameter indicates the number of the row being drawn. The *TextColor* parameter indicates the row's text color, and the *BackColor* parameter indicates the text's background color. If the color is modified within this event, this change is used by the E3Browser when drawing the row. Another important new feature is that if the **GetColumnValue** method is called within this event, returned values are from the drawn row, not the selected row.

4.5.1.3 OnFormatCell

OnFormatCell(*Column*, *FieldName*, *OriginalValue*, *FormattedValue*)

This event allows customizing the format of E3Browser's cells text. Parameters of this event are the following:

- **Column**: Index of E3Browser's visible column (starting at 0). It allows identifying the cell's column being formatted
- **FieldName**: A text with the name of the column's field being formatted
- **OriginalValue**: Cell's unformatted **Variant**-type value
- **FormattedValue**: A formatted **Variant**-type value, according to the configuration of E3Browser's column. If it is modified inside the event, then it allows changing the formatted value

Example (formatting Alarm fields):

```
Sub E3Browser1_OnFormatCell(Column, FieldName, OriginalValue,
FormattedValue)
  If Column = 15 Then
    If Not IsNull(OriginalValue) Then
      FormattedValue = SourceTypeName(OriginalValue)
  ElseIf Column = 9 Then
    If OriginalValue = 0 Then
      FormattedValue = "High"
    ElseIf OriginalValue = 1 Then
      FormattedValue = "Medium"
    Else
      FormattedValue = "Low"
    End If
  End If
End Sub
```

4.5.1.4 MouseMove

MouseMove()

Occurs when the mouse pointer is moved over the E3Browser.

4.5.2 Methods

This section contains information about the methods of the E3Browser object.

4.5.2.1 AboutBox

AboutBox()

This method displays a dialog box with information about E3Browser's version and copyright.

4.5.2.2 ClearFields

ClearFields()

Clears E3Browser's column and row format.

4.5.2.3 GetColumnValue

GetColumnValue(*Index*)

Returns a cell value, in the informed column and selected row. This method has the *Index* parameter, determining the index of the desired column. Example:

```
Sub E3Browser1_DblClick()
  Screen.Item("Text1").Value = GetColumnValue(0)
  Screen.Item("Text2").Value = GetColumnValue(1)
  Screen.Item("Text3").Value = GetColumnValue(2)
  Screen.Item("Text4").Value = GetColumnValue(3)
End Sub
```

4.5.2.4 Requery

Requery()

The **Requery** method updates the Query, by using its current settings, and then returns data to the E3Browser.

4.5.2.5 RetrieveE3QueryFields

RetrieveE3QueryFields()

The **RetrieveE3QueryFields** method reads a Query data structure and updates the E3Browser format with the fields defined by that Query. If it is successful, it returns

True. Otherwise, it returns False. This method is especially useful when users must use a single E3Browser to display data from different tables or queries.

4.5.3 Properties

This section contains information about the properties of the E3Browser object.

4.5.3.1 AllowColumnResize

◩ Enables or disables E3Browser's grid column size configuration, at run time. If False, column size is fixed, and cannot be modified.

4.5.3.2 AllowRowResize

◩ Enables or disables E3Browser's grid row size configuration, at run time. If False, row size is fixed, and cannot be modified.

4.5.3.3 ColumnWidth

9 Determines E3Browser's column width, in pixels.

4.5.3.4 CurSel

9 Indicates E3Browser cursor's current position, that is, the row index where cursor is positioned.

4.5.3.5 E3Query

🔗 Returns E3Browser's Query object, so that its properties can be accessed.

4.5.3.6 Fields

🔗 Returns the Collection that contains a list of all table fields, allowing its reference through the items of this collection. Default value is empty. Example:

```
Sub E3Browser1_Click()
  ' Changes the color of Field1
  Set fields = Screen.Item("E3Browser").Fields
  Set Field1 = fields.Item("Field1")
  field1.BkColor = RGB(255, 0, 0) ' Red
  ' Shows how many fields E3Browser has
  MsgBox fields.Count
  ' Shows E3Browser's number of fields
  For Each Fields In fields
    MsgBox fields.Name
  Next
End Sub
```

4.5.3.7 FixedBkColor

Specifies E3Browser's first column background color. Default value is beige (**RGB(236, 233, 216)**).

4.5.3.8 FixedColumnWidth

Specifies E3Browser's first column width (in pixels). Default value is 30.

4.5.3.9 FixedRowFont

Determines the font to be used at E3Browser's header row. This property cannot be used in scripts or Links, and is configured only via Studio. Default value is "Arial".

4.5.3.10 FixedRowHeight

Determines E3Browser's header row height (in pixels). Default value is 20.

4.5.3.11 FixedTextColor

Changes E3Browser's header color.

4.5.3.12 GridBkColor

Determines E3Browser's data area background color. Default value is white (**RGB(255, 255, 255)**).

4.5.3.13 GridFont

Determines the font to be used on E3Browser's data area texts. Default value is "Arial". This property cannot be used in scripts or Links, being configured only via Studio.

4.5.3.14 GridLineColor

Determines E3Browser's data grid line color. Default value is gray (**RGB(192, 192, 192)**).

4.5.3.15 GridLinesType

Determines the type of rows to be drawn on E3Browser's data grid.

Available options for GridLinesType

OPTION	DESCRIPTION
0 - GLNone	No grid lines.
1 - GLHorz	Only horizontal grid lines (default).
2 - GLVert	Only vertical grid lines.
3 - GLBoth	Horizontal and vertical grid lines.

4.5.3.16 RefreshTime

Specifies Query's refresh time, relative to the specific database. Through this property it is possible to verify data update at the related Historic referring to a specific time (in milliseconds). When the **RefreshTime** property is equal to 0, there is no data update, therefore data remains unchanged.

4.5.3.17 RowHeight

Defines E3Browser's rows height, in pixels. Default value is 20.

4.5.3.18 SelectRow

Determines whether it is possible to select E3Browser's rows. If True, these rows can be selected. Otherwise, row selection is disabled.

4.5.3.19 SourceQuery

Contains a reference to an E3Query object connected to this E3Browser.

NOTE: To change E3Browser's Query via scripts (in case the new Query modifies the original Query fields), in addition to changing the **SourceQuery** property, users must use the **RetrieveE3QueryFields** and **Requery** methods.

4.5.3.20 TextBkColor

Specifies E3Browser's data cells background color. Default value is white (**RGB(255, 255, 255)**).

4.5.3.21 TextColor

Specifies E3Browser's text color. Default value is black (**RGB(0, 0, 0)**).

4.5.3.22 TitleTipBkColor

Specifies E3Browser's tip background color. Default value is black (**RGB(0, 0, 0)**).

4.5.3.23 TitleTipTextColor

Specifies E3Browser's tip text color. Default value is gray (**RGB(204, 204, 204)**).

4.5.3.24 ToolbarBkColor

Specifies E3Browser's toolbar background color. Default value is beige (**RGB(236, 233, 216)**).

4.5.3.25 ToolbarFont

Determines E3Browser's toolbar text font. This property cannot be used in scripts or Links, being configured only via Studio.

4.5.3.26 ToolbarForeColor

Specifies E3Browser's toolbar foreground color. Default value is black (**RGB(0, 0, 0)**).

4.5.4 E3Browser Fields

This section contains information about **properties** of the E3Browser Field object. This object does not have events nor methods associated to it.

4.5.4.1 Properties

This section contains information about the properties of the E3Browser field object.

4.5.4.1.1 BkColor

Determines E3Browser field's background color. The default value is the color configured on Windows for the **Window** item on **Control Panel (Control Panel - Display - Appearance - Advanced)**.

4.5.4.1.2 Color

Returns the Field's text color. The default value of this property is black (**RGB(0, 0, 0)**).

4.5.4.1.3 Format

A Configures the formatter used on the Field column.

4.5.4.1.4 Name

A Returns the Field's name.

4.5.4.1.5 Visible

◩ Enables or disables the visibility of the selected Field on E3Browser's Query. If this property is configured to True, this Field is visible on E3Browser. Otherwise, this Field is not shown on E3Browser at run time. The default value is True.

4.5.4.1.6 Width

9 Returns the Field's width, in pixels.

4.6 E3Chart

This section contains information about **events**, **methods**, and **properties** of the E3Chart object.

4.6.1 Events

This section contains information about the events of the E3Chart object.

4.6.1.1 OnCursorChange

OnCursorChange()

This event occurs whenever an E3Chart cursor changes its position. For example, users can create a script for this event when they need to show cursor position values on a Screen.

```
Sub E3Chart1_OnCursorChange()
  Set Chart = Application.GetFrame("").Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.Item(0)
  ' The Text1 object must display the current position of the
cursor
  Set Text = Application.GetFrame("").Screen.Item("Text1")
  If Pen.GetCursorPos(aa, bb) Then
    Text.Value = "X Position = " & aa & "; Y Position = " & bb
  End If
End Sub
```

4.6.1.2 OnLegendClick

OnLegendClick(*Row*, *Col*, *RowData*)

Occurs whenever users click one of the Legend rows. The *Row* and *Col* parameters indicate, respectively, the row and column clicked. The *RowData* parameter is the Legend's Pen index where the click occurred. Example:

```
Sub E3Chart1_OnLegendClick(Row, Col, RowData)
  Set text = Screen.Item("Text1")
  text.Value = Legend.Item(col).Name & " " & _
    Pens.Item(RowData).name
End Sub
```

4.6.1.3 OnQueryFinish

OnQueryFinish()

Occurs whenever one or more Queries are finished. When this event is generated, the call for the **FitAll** or **FitPen** methods may cause problems if users are using automatic Queries, since these methods activate other Queries until all data is read. In this case, it is recommended that the value passed by these methods' parameters be 1 (one), which fits Pens vertically.

4.6.2 Methods

This section contains information about the methods of the E3Chart object.

4.6.2.1 ClearPenMarks

ClearPenMarks()

Removes the cursor marks from all E3Chart Pens.

4.6.2.2 CopyConfig

CopyConfig(*SourceChart*[, *Flags*])

The **CopyConfig** method copies settings from an E3Chart to another one. The *SourceChart* parameter indicates the source E3Chart, whose properties are copied to the E3Chart calling this method.

NOTE: In case of Reports, the **CopyConfig** method works only with **Historic**-type Pens.

For example, consider a copy of settings from an E3Chart on a Screen (in this example, `ScreenChart`) to another one within a Report (`ReportChart`). In this case, the script must be added to the E3Report object associated to the Report.

```
Sub OnBeforePrint
```

```
  Set Chart = _
    Report.Sections("PageHeader").Controls("ReportChart")
  Chart.CopyConfig(Application.GetFrame()._
    Screen.Item("ScreenChart"))
  Chart.LoadData()
  Chart.FitAll()
End Sub
```

NOTE: This method also has an optional and not used parameter called *Flags*, kept only for compatibility reasons.

4.6.2.3 FitAll

FitAll([*FitStyle*])

Fits all Pens into the E3Chart. The optional parameter *FitStyle* indicates how Pens fit at run time:

- **0**: Fits both Axes at the same time
- **1**: Fits only the Vertical Axis
- **2**: Fits only the Horizontal Axis

Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").FitAll()
End Sub
```

4.6.2.4 FitPen

FitPen(*Pen*[, *FitStyle*])

Fits a Pen into the E3Chart specified by index or name. The *Pen* parameter defines the Pen to be fit into the E3Chart. The optional parameter *FitStyle* indicates how Pens fit at run time:

- **0**: Fits both Axes at the same time
- **1**: Fits only the Vertical Axis
- **2**: Fits only the Horizontal Axis

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.FitPen(1)
  Chart.FitPen("Pen1", 1)
  ' Fits Pen1 only vertically
End Sub
```

4.6.2.5 LoadData

LoadData()

Loads data into an E3Chart. This method is specially used for loading data before printing, when used in a Report object. Example:

```
Sub CommandButton1_Click()
  MsgBox Screen.Item("E3Chart1").LoadData()
End Sub
```

NOTE: The **LoadData** method is synchronous only if the Pen is not in **Automatic** mode.

4.6.2.6 ResetConfig

ResetConfig([*Flags*])

Removes all configurations set in an E3Chart, rolling them back to their initial status. Example:

```
Sub E3Chart1_OnStartRunning()
  ' When starting E3Chart1, removes all settings
  ResetConfig()
End Sub
```

NOTE: This method also has an optional parameter called *Flags*, which is kept only for compatibility reasons.

4.6.2.7 ShowCursors

ShowCursors()

Activates the **Interval Search** mode. At run time, this feature can be accessed by right-clicking the object and then selecting the **Interval Search** option on the contextual menu.

4.6.2.8 ZoomIn

ZoomIn()

The **ZoomIn** method increases zoom in the E3Chart, that is, it closes up Pen viewing. At run time, this feature can be accessed by right-clicking the object and selecting the **Zoom In** option on the contextual menu. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").ZoomIn()
End Sub
```

4.6.2.9 ZoomOut

ZoomOut()

The **ZoomOut** method decreases zoom in the E3Chart, that is, it opens up Pen viewing in E3Chart. At run time, this feature can be accessed by right clicking the E3Chart and selecting the **Zoom Out** option on the contextual menu. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").ZoomOut()
End Sub
```

4.6.3 Properties

This section contains information about the properties of the E3Chart object.

4.6.3.1 Axes

Returns the E3Chart's Axes Collection. Then, the collection properties can be modified.

4.6.3.2 BackColor

Determines the E3Chart's background color. For this color to be displayed, the **ShowBackground** property must be set to True. Default value is beige (**RGB(236, 233, 216)**).

4.6.3.3 CursorBegin

Sets the initial cursor position, between 0 and 1. Either the **ShowCursors** method must be executed, or the **Interval Search** option must be enabled for the cursor to appear.

4.6.3.4 CursorColor

Determines the color of the interval search cursor. Default value is red (**RGB(255, 0, 0)**).

4.6.3.5 CursorEnd

Sets the final cursor position, between 0 and 1. Either the **ShowCursors** method must be executed, or the **Interval Search** option must be enabled for the cursor to appear.

4.6.3.6 CursorLineStyle

Establishes the line style for interval search cursor. The available options are:

Available options for CursorLineStyle

OPTION	DESCRIPTION
0 - LS_Solid	Applies a solid line in the E3Chart's interval cursor.
1 - LS_Dash	Applies a dashed line in the E3Chart's interval cursor.
2 - LS_Dot	Applies a dotted line in the E3Chart's interval cursor.
3 - LS_Dashdot	Applies a dash-dotted line in the E3Chart's interval cursor.
4 - LS_Dashdotdot	Applies a dash-dot-dot line in the E3Chart's interval cursor.
5 - LS_Null	Applies an invisible line in the E3Chart's interval cursor.

4.6.3.7 CursorLineWidth

Establishes the cursor line width.

4.6.3.8 CursorSearchStyle

Allows the cursor to search for chart points, according to the following options:

- **0 - PointNearest**: Searches for the nearest point
- **1 - LinearInterpolation**: Searches for an interpolated point
- **2 - PointPrevious**: Searches for the previous point

4.6.3.9 ForeColor

Determines the E3Chart's foreground color. The default value of this property is black (**RGB(0, 0, 0)**).

4.6.3.10 GridBkColor

Determines the E3Chart grid's background color. The default value of this property is white (**RGB(255, 255, 255)**). Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.GridBkColor
  MsgBox "Next"
  E3Chart1.GridBkColor = RGB(0, 255, 0)
  MsgBox "Return"
```

```
  E3Chart1.GridBkColor = Old
End Sub
```

4.6.3.11 HorAxisTitle

Determines the title of the main horizontal Axis. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1= Screen.Item("E3Chart1")
  Old = E3Chart1.HorAxisTitle
  MsgBox "Next"
  E3Chart1.HorAxisTitle = "Test"
  MsgBox "Return"
  E3Chart1.HorAxisTitle = Old
End Sub
```

4.6.3.12 HorGrid

Determines the line type to be applied on the E3Chart's horizontal grid.

Available options for HorGrid

OPTION	DESCRIPTION
0 - Solid	Applies a solid line on the E3Chart's horizontal grid.
1 - Dash	Applies a dashed line on the E3Chart's horizontal grid.
2 - Dot	Applies a dotted line on the E3Chart's horizontal grid (default).
3 - Dashdot	Applies a dash-dot line on the E3Chart's horizontal grid.
4 - Dashdotdot	Applies a dash-dot-dot line on the E3Chart's horizontal grid.
5 - Invisible	Applies an invisible line on the E3Chart's horizontal grid.

Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorGrid
  For i = 0 To 5
    E3Chart1.HorGrid = i
    MsgBox "E3Chart1.HorGrid =" & CStr(i)
  Next
  MsgBox "Return"
  E3Chart1.HorGrid = Old
End Sub
```

4.6.3.13 HorGridColor

Determines the color of the E3Chart's horizontal grid. The default value of this property is gray (**RGB(192, 192, 192)**). Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorGridColor = RGB(255, 0, 0)
  MsgBox "Next"
  E3Chart1.HorGridColor = RGB(255, 0, 0)
  MsgBox "Next"
  E3Chart1.HorGridColor = RGB(0, 0, 255)
  MsgBox "Return"
  E3Chart1.HorGridColor = Old
End Sub
```

4.6.3.14 HorMinorTicks

Determines the number of subdivisions of the grid horizontal scales. The default value of this property is 1. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorMinorTicks
  For i = 0 To 5
    E3Chart1.HorMinorTicks = i
    MsgBox "Next value"
  Next
  E3Chart1.HorMinorTicks = Old
End Sub
```

4.6.3.15 HorScaleBegin

Determines the initial value to be applied on the grid's main horizontal scale. This value can either be numerical (for XY E3Charts) or data (for fixed time scale E3Charts). For real time E3Charts this property is not applied, and users must use the **TimeSpan** property instead.

4.6.3.16 HorScaleEnd

Determines the final value to be applied on the grid's main horizontal scale. This value can either be numerical (for XY E3Charts) or data (for fixed time scale E3Charts). For real time E3Charts this property is not applied, and users must use the **TimeSpan** property instead. Example:

```
Sub ComboBox1_Change()
  ' Defines which query to be displayed
  current_query_index = ListIndex
  Set E3Chart1 = Screen.Item("E3Chart1")
  i = 0
  For Each query In E3Chart1.Queries
```

```
    If i = current_query_index Then
      ' Retrieves everything
      query.FieldFilter(0) = ""
      Set current_query = query
    Else
      ' Retrieves everything
      ' to prevent slowing down the operation
      query.FieldFilter(0) = "<0"
    End If
    i = i + 1
  Next
  ' Only displays pens which use the current query
  For Each pen In E3Chart1.Pens
    pen.Visible = (pen.QueryName = current_query.Name)
  Next
  ' Updates queries
  E3Chart1.Queries.UpdateData()
  Screen.Item("E3Chart1").HorScaleBegin = Now - 0.001
  Screen.Item("E3Chart1").HorScaleEnd = Now
End Sub
```

4.6.3.17 HorScaleFormat

A Contains a text that represents a mask where object values are displayed. This mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for displaying negative numbers as signed or between parentheses. For very large or very small numbers, the **Scientific** format is recommended
- **Date**: Displays numerical date values (when valid). To represent only time, use the equivalent format
- **Time**: Displays numerical time values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies the number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays the number with a mantissa and exponent notation. Ideal for numbers with variable magnitude. Allows up to 15 decimal places
- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal, or binary, for example)
- **Other**: Allows directly editing the format code, or selecting a previously created format

The mask for these formats, as specified in the **Type** field, is displayed in the Properties window (for example, "M/d/yy", "0E-00", etc.).

4.6.3.18 HorTickUnit

Determines the number of subdivisions between grid marks. When this property is equal to 0, spacing is automatic. Example:

```
Sub SubCommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorTickUnit
  For i = 0 To 30 Step 10
    E3Chart1.HorTickUnit = i
    MsgBox "E3Chart1.HorTickUnit = " & CStr(i)
  Next
  MsgBox "Return"
  E3Chart1.HorTickUnit = Old
End Sub
```

4.6.3.19 Legend

Returns the E3Chart's Legend object. Then, Legend properties can be changed.

4.6.3.20 MouseMode

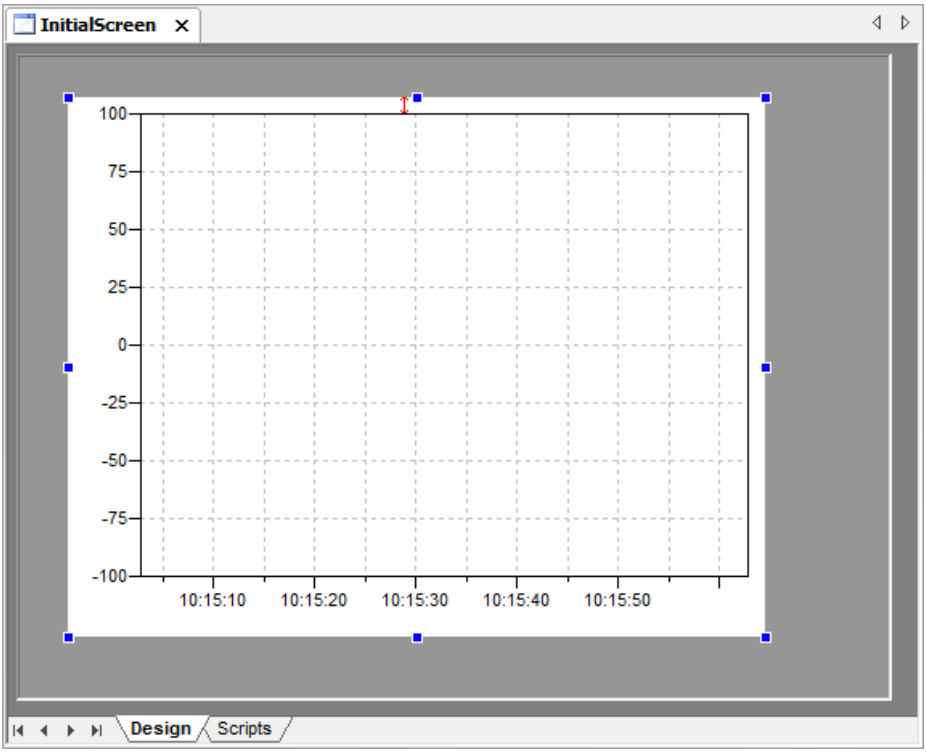
Selects one of the runtime options of the E3Chart's contextual menu. The available options for this property are:

- **0 - MouseModeZoom**: Puts mouse in **Zoom** mode by selected area. This option is available in numerical scale charts in XY and fixed scale. This is the equivalent of selecting the **Zoom Box** option of the E3Chart's contextual menu
- **1 - MouseModePan**: Puts mouse in **Scale moving** mode. This is the equivalent of selecting the **Move** option of the E3Chart's contextual menu
- **2 - MouseModePanH**: Puts mouse in **Scale moving** mode only in the horizontal direction. This is the equivalent of selecting the **Move horizontally** option of the E3Chart's contextual menu
- **3 - MouseModeSearch**: Puts mouse in **Pen data values search** mode. This is the equivalent of selecting the **Search** option of the E3Chart's contextual menu
- **4 - MouseModeCursors**: Enables the **Time interval search** option. This is the equivalent of selecting the **Search Intervals** option of the E3Chart's contextual menu

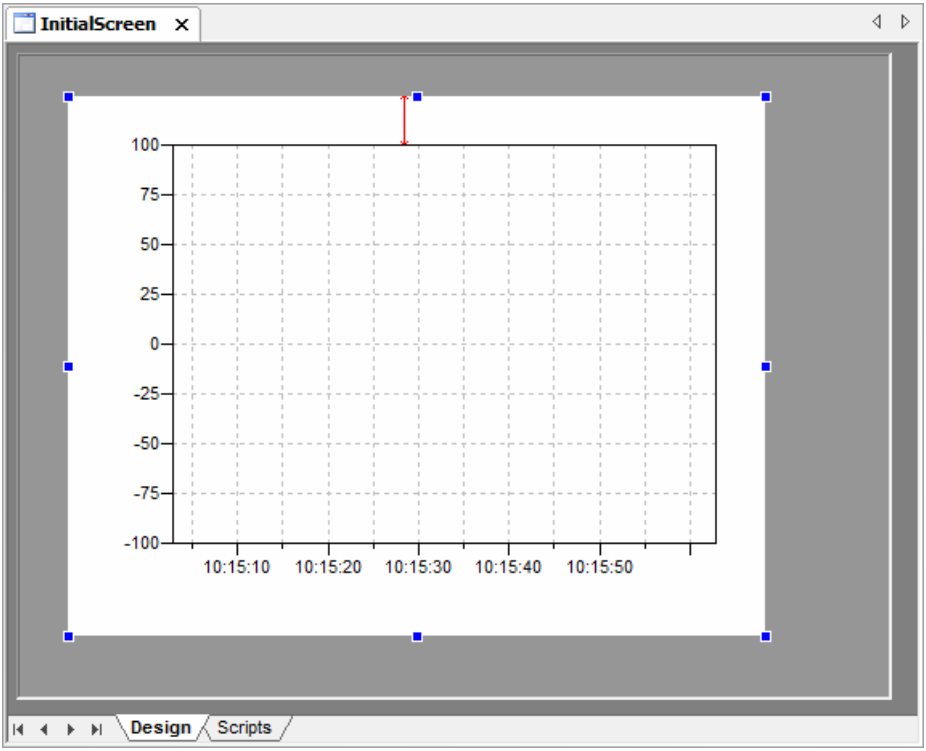
4.6.3.21 Padding

This property determines the distance, in pixels, between the chart and the border of the E3Chart, as seen on the next figures with the red arrows. The default value of

this property is 10. Example:



Padding property equal to 10



Padding property equal to 30

4.6.3.22 Pens

Returns the E3Chart's Pen Collection object. The Pen Collection object is used to insert, remove, or access the available Pens of an E3Chart. This is a read-only property. Example:

```
Sub CommandButton1_Click()
  For Each pen In Screen.Item("E3Chart1").Pens
    pen.Visible = True
  Next
End Sub
```

4.6.3.23 Queries

Returns the E3Chart's Query Collection object. The Query Collection object is used to insert, remove, or access the available Queries of an E3Chart. This is a read-only property.

4.6.3.24 RefreshTime

This property determines the E3Chart's update time.

4.6.3.25 ScaleFont

A Determines the text font used on the grid. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").ScaleFont = "Times New Roman"
  Screen.Item("E3Chart1").ScaleFont.Size = 12
  Screen.Item("E3Chart1").ScaleFont.Italic = True
End Sub
```

4.6.3.26 ShowBackground

◩ Enables or disables viewing the chart background. If this property is True, the chart background is displayed. Otherwise, the chart has a transparent background. The color chosen in the **BackColor** property does not appear if this property is False (default). Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.ShowBackground = Not E3Chart1.ShowBackground
End Sub
```

4.6.3.27 ShowBottomScale

◩ If this property is set to True, the main horizontal Axis is displayed on the grid bottom. Otherwise, it is not displayed. The default value is True. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  oldBottomScale = E3Chart1.ShowBottomScale
  MsgBox "Display axis"
  E3Chart1.ShowBottomScale = True
  MsgBox "Hide axis"
  E3Chart1.ShowBottomScale = False
  MsgBox "Return..."
  E3Chart1.ShowBottomScale = oldBottomScale
End Sub
```

4.6.3.28 ShowGridBackground

◩ Enables or disables viewing the grid background. If this property is True (default), the grid background is displayed. Otherwise, the grid has a transparent background. The chosen color in the **GridBkColor** property does not appear if this property is set to False. Example:

```
Sub CommandButton1_Click()
  Set Chart1 = Screen.Item("E3Chart1")
  Chart1.ShowGridBackground = Not Chart1.ShowGridBackground
End Sub
```

4.6.3.29 ShowLeftScale

◩ If this property is set to True, the main vertical Axis is displayed on the left side of the grid. Otherwise, it is invisible. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.ShowLeftScale = Not Chart.ShowLeftScale
End Sub
```

4.6.3.30 ShowRightScale

◩ If this property is set to True, the main vertical Axis is displayed on the right size of the grid. Otherwise, it is invisible. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.ShowRightScale = Not Chart.ShowRightScale
End Sub
```

4.6.3.31 ShowPopupMenu

◩ Enables or disables the option of displaying the E3Chart's contextual menu. If the value is equal to True, this menu is displayed when the user right-clicks the E3Chart's chart. If the value is False, this menu is not displayed. The default value of this property is True.

4.6.3.32 ShowTitle

◩ If this property is set to True, the E3Chart's main title is visible. Otherwise, it is invisible. The **Title** property contains the title to be displayed in the E3Chart. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  oldTitle = E3Chart1.Title
  oldShowTitle = E3Chart1.ShowTitle
  E3Chart1.Title = "Test!"
  MsgBox "Display"
  E3Chart1.ShowTitle = True
  MsgBox "Hide"
  E3Chart1.ShowTitle = False
  MsgBox "Return"
  E3Chart1.Title = oldTitle
  E3Chart1.ShowTitle = oldShowTitle
End Sub
```

4.6.3.33 ShowTopScale

If this property is set to True, the main horizontal Axis on top of the grid. Otherwise, it is not displayed. The default value is False. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.ShowTopScale = Not Chart.ShowTopScale
End Sub
```

4.6.3.34 TimeSpan

Indicates the time scale on E3Chart's main horizontal Axis, when it is configured to show a real time scale. The value of this property is always in seconds. The default value is 60. Example:

```
Sub RoundRect1_Click()
  MsgBox Screen.Item("E3Chart1").TimeSpan
End Sub
```

4.6.3.35 Title

Determines the E3Chart's main title. For this title to appear on the E3Chart, the **ShowTitle** property must be set to True.

4.6.3.36 TitleFont

Determines the E3Chart's main title font. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.Title = "Test"
  E3Chart1.ShowTitle = True
  MsgBox "Change font"
  E3Chart1.TitleFont = "Times New Roman"
  MsgBox "Change size"
  E3Chart1.TitleFont.Size = 20
End Sub
```

4.6.3.37 VerAxisTitle

Determines the title of the main vertical Axis. Example:

```
Sub ComboBox1_Change()
  Screen.Item("E3Chart1").VerAxisTitle = "Title1"
End Sub
```

4.6.3.38 VerGrid

Determines the line type applied on the E3Chart's vertical grid.

Available options for VerGrid

OPTION	DESCRIPTION
0 - Solid	Applies a solid line on the E3Chart's vertical grid.
1 - Dash	Applies a dashed line on the E3Chart's vertical grid.
2 - Dot	Applies a dotted line on the E3Chart's vertical grid (default).
3 - Dashdot	Applies a dash-dot line on the E3Chart's vertical grid.
4 - Dashdotdot	Applies a dash-dot-dot line on the E3Chart's vertical grid.
5 - Invisible	Applies an invisible line on the E3Chart's vertical grid.

Example:

```
Sub E3Chart1_OnStartRunning()
  VerGrid = 2
End Sub
```

4.6.3.39 VerGridColor

Determines the line color of the grid's main vertical Axis. The default value of this property is gray (**RGB(192, 192, 192)**). Example:

```
Sub RoundRect1_Click()
  Screen.Item("E3Chart1").VerGridColor = RGB(255, 0, 0)
End Sub
```

4.6.3.40 VerMinorTicks

Determines the number of subdivisions between marks of the grid's main vertical Axis. The default value of this property is 1. Example:

```
Sub RoundRect1_Click()
  Screen.Item("E3Chart1").VerMinorTicks = 3
End Sub
```

4.6.3.41 VerScaleBegin

Determines the value on top of the grid's main vertical Axis. The default value of this property is 100. Example:

```
Sub RoundRect1_Click()
  MsgBox Screen.Item("E3Chart1").VerScaleBegin
```

```
End Sub
```

4.6.3.42 VerScaleEnd

9 Determines the value on bottom of the grid's main vertical Axis. The default value of this property is -100. Example:

```
Sub RoundRect1_Click()
  MsgBox Screen.Item("E3Chart1").VerScaleEnd
End Sub
```

4.6.3.43 VerScaleFormat

A Contains a text that represents a mask where object values are displayed. This mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for displaying negative numbers as signed or between parentheses. For very large or very small numbers, the **Scientific** format is recommended
- **Date**: Displays numerical date values (when valid). To represent only time, use the equivalent format
- **Time**: Displays numerical time values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies the number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays the number with a mantissa and exponent notation. Ideal for numbers with variable magnitude. Allows up to 15 decimal places
- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal, or binary, for example)
- **Other**: Allows directly editing the format code, or selecting a previously created format

The mask for these formats, as specified in the **Type** field, is displayed in the Properties window (for example, "M/d/yy", "0E-00", etc.).

4.6.3.44 VerTickUnit

9 Determines the number of subdivisions between grid marks. When this property is set to 0, spacing is automatic. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
```

```
  Old = E3Chart1.VerTickUnit
  For i = 0 To 30 Step 10
    E3Chart1.VerTickUnit = i
    MsgBox "E3Chart1.VerTickUnit = " & CStr(i)
  Next
  MsgBox "Return"
  E3Chart1.VerTickUnit = Old
End Sub
```

4.6.4 Pen Collection

This section contains information about **methods** and **properties** of the Pen Collection object. This object does not have events associated to it.

4.6.4.1 Methods

This section contains information about the methods of the Pen Collection object.

NOTE: The E3Chart pen collection must be accessed via **Pens** property.

4.6.4.1.1 AddPen

AddPen(*PenName*)

The **AddPen** method adds a new Pen to the E3Chart, returning the Pen created. Example:

```
Sub CommandButton1_Click()
  ' Creates an untitled pen.
  Set Pen = Screen.Item("E3Chart1").Pens.AddPen("")
  MsgBox Pen.Name
End Sub

SubCommandButton1_DbClick()
  ' Creates a pen named "Pen1".
  ' If the name already exists, increments it.
  Set Pen = Screen.Item("E3Chart1").Pens.AddPen("Pen1")
  MsgBox Pen.Name
End Sub

Sub CommandButton2_Click()
  ' Creates a pen and links it to DemoTag1.
  Set Chart = Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.AddPen("")
  MsgBox Pen.Name
  Pen.UsetimeStamp = True
  Pen.YLink = "Data.DemoTag1"
  Pen.Connect()
End Sub
```

4.6.4.1.2 ChangePenPos

ChangePenPos(*Source*, *Dest*)

Modify the drawing position of Pens in an E3Chart. This method is especially useful when a Pen has a line shape, and another one has an area shape. If the area Pen is drawn after the line Pen, it may hide the last one. One solution could be to invert the drawing order of these Pens.

- **Source**: Determines the index of the Pen to be moved (starting at 1)
- **Dest**: Determines the Pen destination (starting at 1)

Example:

```
Sub CommandButton1_Click()
  ' Moves pen 1 to position 2.
  Screen.Item("E3Chart1").Pens.ChangePenPos(1, 2)
End Sub
```

4.6.4.1.3 Item

Item(*Index*)

The **Item** method turns a Pen object back to the Pen Collection, specified by an index. This method has the *Index* parameter that can be a numeric type if corresponding to the Pen's index, or a text if it corresponds to the Pen's name. Example:

```
Sub CommandButton1_Click()
  ' Gets the first pen.
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item(0)
End Sub
```

4.6.4.1.4 Remove

Remove(*Name*)

The **Remove** method removes a Pen using the specified name. This method has a *Name* parameter determining the name of the Pen to be removed. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").Pens.Remove(1)
End Sub
```

4.6.4.1.5 SetCursorPos

SetCursorPos(*X*, *Range*)

Positions the cursor of every E3Chart Pen. It is the same behavior as the **SetCursorPos** method of each Pen.

4.6.4.2 Properties

This section contains information about the properties of the Pen Collection object.

4.6.4.2.1 Count

9 Contains the total amount of Pens inserted on the E3Chart. This is a read-only property.

4.6.4.3 Pens

This section contains information about **methods** and **properties** of the Pen object. This object does not have events associated to it.

4.6.4.3.1 Methods

This section contains information about the methods of the Pen object.

4.6.4.3.1.1 AddPoint

AddPoint(*ValueX*, *ValueY*[, *Quality*])

Adds a point at the end of the real time buffer. This buffer size is only valid after connecting the Pen. If the Pen is created in Studio, connection is automatic, but if it is created via script, it is necessary to call the **Connect** method after creating it. This method must be used with real time Pens, and with the **UseTimeStamp** property set to False. The optional parameter *Quality* indicates the quality of the point to be inserted. If this parameter is not informed, the point quality is considered as Good (192). The number of points that can be inserted into a Pen depends on the buffer size (the Pen's **BufferSize** property).

4.6.4.3.1.2 Clear

Clear()

Erases data from real time buffers, without decreasing its size. This method does not disconnect Links, nor removes historical data.

4.6.4.3.1.3 Connect

Connect()

The **Connect** method connects the Pen to the server to receive data in real time, associating the **XLink** and **YLink** properties. If the Pen is already connected, this method remains inactive. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
```

```
  Pen1.Disconnect()
  Pen1.Connect()
End Sub
```

4.6.4.3.1.4 Disconnect

Disconnect()

The **Disconnect** method cleans up current data and prevents a Pen from receiving more real time data from its associated Tag. If the Pen is already disconnected, this method does nothing. It can be used to change a Tag at run time. When the **Disconnect** method is used with a mixed Pen (the **DataSourceType** property equal to 2), this method removes the real time part, and keeps the historic part. At run time, the **Connect** method must be called to show real time data again. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Disconnect()
  Pen1.Connect()
End Sub
```

4.6.4.3.1.5 GetCursorPos

GetCursorPos(*X*, *Y*)

Returns to the position where the cursor intercepted the E3Chart's Pen. This method has the *X* and *Y* parameters, corresponding to x and y cursor positions. If this method is successful, it returns True, otherwise it returns False. Example:

```
Sub CommandButton1_Click()
  For Each pen In Chart.Pens
    If pen.GetCursorPos(aa, bb) Then
      strResult = strResult & pen.name & _
        " := " & CSTr(CDate(aa)) + _
        "y " + CStr(bb) + vbNewLine
    End If
  Next
  MsgBox strResult
End Sub
```

4.6.4.3.1.6 GetPoint

GetPoint(*ValueX*, *ValueY*)

Returns the X and Y coordinates of the nearest point of the input value in the *ValueX* parameter. The *ValueX* parameter informs the reference value to search for the point, and then gets the effective value of the X coordinate of the nearest point found. The *ValueY* parameter returns the effective value of the Y coordinate of the nearest point found. This method returns True if a point is found, and False otherwise.

4.6.4.3.1.7 SetCursorPos

SetCursorPos(*X*, *Range*)

Positions the cursor of the E3Chart Pen. The *X* parameter indicates what is the position where the cursor must be positioned, similar to the behavior of moving the cursor in a mouse search. The cursor is moved to the nearest position indicated by the *X* parameter. The *Range* parameter is optional and used as a validation. The cursor is only moved if the valid point is inside the interval. Any negative value indicates that the passed interval must not be used. Example:

```
' If there is a valid point in (x = 1, y = 10)
' and another one in (x = 4, y = 20)
SetCursorPos(2) ' moves the cursor to the point (1, 10)
SetCursorPos(4) ' moves the cursor to the point (4, 20)

' When range is used, the cursor is only moved
' if the point is inside the range.
' Does not move the cursor, because 2 is more than 0.5 units
' away from the nearest point, which is 1
SetCursorPos (2, 0.5)
' Sends the cursor to the point (1, 10)
SetCursorPos (2, 4)
```

This method returns True if the cursor moves, otherwise returns False.

4.6.4.3.2 Properties

This section contains information about the properties of the Pen object.

4.6.4.3.2.1 AutoQuery

◩ When **AutoQuery** property is set to True, one says that this is an **AutoQuery**-type Pen, or simply an **Automatic Pen**. The goal of an Automatic Pen is to save memory consumption and query time. To do so, it applies filters on **E3Timestamp** field to bring only needed data into E3Chart's area. Every time an E3Chart's visible period changes, an Automatic Pen retrieves missing data to complete the drawing of that period. In addition, an AutoQuery also completes historical missing data to connect Pen's historical and real time parts. If, by any reason, historical data is not retrieved after thirty seconds, an AutoQuery is cancelled on that part. See also the **MaxGapTime** property for more details.

Due to the way an AutoQuery applies filters to an **E3TimeStamp** field, it is not available for user-defined Storage queries or SQL code. That is, even when **AutoQuery** property is set to True, it has no effect on **Storage**-type Queries. One way to recognize an AutoQuery is a hatched drawing on the E3Chart screen. Every time hatches appear on that drawing, this means that on this period an AutoQuery is being performed. When hatch borders are in red, this means that AutoQuery on that

part is on failure. In this case, E3Chart reruns problematic Queries on that part.

NOTE: Different from a real time part, where each Pen has its own data buffer, Pen's historical part is stored on the Query, and it is shared among Pens. For example, when a Query has three fields, **E3Timestamp**, **Field1**, and **Field2**, this data is stored on the Query and it is available for Pens sharing that Query. Thus, the common part, usually the **E3Timestamp** field, can be used by two different Pens, without duplicating data. In case of Automatic Pens, two different Pens can use the same table and, due to different scales, they can load different Query periods. In this situation, each Pen automatically inherits the part loaded by the other Pen.
The Query object cannot work simultaneously in **Automatic** and **Non-Automatic** mode. This also means that if different Pens, one Automatic and another one Non-Automatic, want to share the same Query, that Query is going to adapt to the first Pen using it. That is, the **AutoQuery** property does not ensure that a Query is in **Automatic** mode, this depends on other factors.

4.6.4.3.2.2 AverageY

Informs the average of the Pen in the interval, in case the **EnableCalc** property is enabled. In case the E3Chart is in **Interval Search** mode, displays the average of this interval. Otherwise, displays the interval between the beginning and the end on the horizontal Axis. Values with bad quality are not considered in case the **ShowBadPoints** is disabled. This is a read-only property.

4.6.4.3.2.3 BkColor

Determines the background color used on a Pen of **Area** type. The default value of this property is empty. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.BkColor = RGB(255, 0, 0)
End Sub
```

4.6.4.3.2.4 BufferSize

Determines the number of points kept on the Pen in real time. After this value, older data is discarded. On historic Pens, this property has no effect. This property is only considered after connecting to the Pen. For more information, see the **Connect** method. The default value of this property is 1000, and it must be always greater than 0. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Disconnect()
  Pen1.BufferSize = 5000
  Pen1.Connect()
End Sub
```

4.6.4.3.2.5 Color

Determines the line color of the Pen in the E3Chart. The default value of this property is empty. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Color= RGB(212, 208, 20)
End Sub
```

4.6.4.3.2.6 DataSourceType

Determines Pen data source. The available options for this property are the following:

Available options for DataSourceType

OPTION	DESCRIPTION
0 - Real Time	Indicates connection of the Pen to a tag updated in real time.
1 - Historic	Indicates connection of the Pen to data from a Query.
2 - Real Time & Historic	Indicates connection of the Pen to real time and historic tags data simultaneously.

When the **DataSourceType** property is equal to 0 (Real Time), the **XLink** and **YLink** properties inform the Links used, or else the **UseTimeStamp** property informs that the **XLink** property is not used, being replaced by the timestamp of the **YLink** property. When the **DataSourceType** property is equal to 1 (Historic), the **XField** and **YField** properties informs field tables to be used. The **QueryName** property indicates the name of the table used. When the **DataSourceType** property is equal to 2 (Real Time & Historic), the 0 and 1 options work simultaneously for the Pen.

NOTE: At run time, when this property changes and the Pen stops displaying real time data, you must call the **Connect** method to display data again.

Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  MsgBox "Click OK to create a pen."
  Set Pen = E3Chart1.Pens.AddPen("Pen1")
  Pen.DataSourceType = 0 ' Real time
  Pen.YLink = "Data.DemoTag1"
  Pen.UseTimeStamp = True ' In X uses timestamp
  Pen.Color = RGB(255, 0, 0)
  Pen.Docstring = "Test"
```

```
  MsgBox "Clique OK to connect."
  Pen.Connect() ' Starts receiving data
  MsgBox "Click OK to fit."
  E3Chart1.FitPen(0)
  MsgBox "Click OK to remove this pen."
  E3Chart1.Pens.Remove(Pen.Name)
End Sub
```

4.6.4.3.2.7 DigitalData

◩ Determines the digital plot style. If this property is set to True, the digital plot style assumes that data changing is in digital form, that is, its value relative to the last one changed instantly. Otherwise, the changing is considered linear and the points are bounded by a line. The default value of this property is True. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.DigitalData = True
End Sub
```

4.6.4.3.2.8 EnableCalc

◩ Enables or disables calculating average, minimum, and maximum inside the interval.

4.6.4.3.2.9 EnableHighLimit

◩ Enables or disables checking the high limit.

4.6.4.3.2.10 EnableLowLimit

◩ Enables or disables checking low limit.

4.6.4.3.2.11 EU

A This property is used to identify the engineering unit represented by the value. For example, degrees, meters, KW/h, etc.

4.6.4.3.2.12 HighlightMaxGapTime

◩ Specifies if the line of the visual connection between the historic and real time part of a mixed Pen (defined in **MaxGapTime**) must have a different color (defined in **MaxGapTimeColor**) and style (defined in **MaxGapTimeStyle**). The default value of this property is False.

4.6.4.3.2.13 HighLimit

9 Determines the high limit alarm.

4.6.4.3.2.14 InterpolatedBeginY

Informs the value of the interpolated point where the initial cursor crosses the Pen. This is a read-only property.

4.6.4.3.2.15 InterpolatedEndY

Informs the value of the interpolated point where the final cursor crosses the Pen. This is a read-only property.

4.6.4.3.2.16 LimitPenBkColor

Determines the background color of the Pen when in alarm.

4.6.4.3.2.17 LimitPenColor

Determines the Pen color when in alarm.

4.6.4.3.2.18 LowLimit

Determines the low limit alarm.

4.6.4.3.2.19 MaxGapTime

Allows specifying a limit time to be considered for a visual connection between the historic and real time part of a mixed Pen. The default value of this property is 0. The value of this property can be modified at run time.

NOTE: This property is available from version 3.5 on. For applications created in prior versions and opened in version 3.5, the value of the property is always 0.

4.6.4.3.2.20 MaxGapTimeColor

Allows configuring the color of the visual connection between the historic and real time part of a mixed Pen, defined in the **MaxGapTime** property. The default value of this property is red (**RGB(255, 0, 0)**).

4.6.4.3.2.21 MaxGapTimeStyle

Specifies the line style of the visual connection between the historic and real time part of a mixed Pen, configured in the **MaxGapTime** property. Possible values for this property are the following:

- **0**: Solid
- **1**: Dashed

- **2**: Dotted
- **3**: Dash - Dot
- **4**: Dash - Dot - Dot
- **5**: Invisible

NOTE: The option **5** (Invisible) of this property can only be selected via scripts.

4.6.4.3.2.22 MaxY

9 Informs the maximum value of the Pen in the interval, in case the **EnableCalc** property is enabled. In case the E3Chart is in **Interval Search** mode, displays the average in this interval. Otherwise, displays the average in the interval between the beginning and the end in the horizontal Axis. Values with bad quality are not considered in case the **ShowBadPoints** property is disabled. This is a read-only property.

4.6.4.3.2.23 MinY

9 Informs the minimum value of the Pen in the interval, in case the **EnableCalc** property is enabled. In case the E3Chart is in **Interval Search** mode, displays the average in this interval. Otherwise, display the average in the interval between the beginning and the end on the horizontal Axis. Values with bad quality are not considered if the **ShowBadPoints** property is disabled. This is a read-only property.

4.6.4.3.2.24 Name

A Determines the name of the Pen. Example:

```
Sub CommandButton1_Click
  Screen.Item("E3Chart1").Pens.Name = "Pen1"
End Sub
```

4.6.4.3.2.25 PenStyle

Determines the Pen line style. The default value of this property is 0 (zero). The available options are the following:

Available options for PenLineStyle

OPTION	DESCRIPTION
0 - LsSolid	Solid line.
1 - LsDash	Dashed line.
2 - LsDot	Dotted line.
3 - LsDashDot	Dash-dot line.
4 - LsDashDotDot	Dash-dot-dot line.
5 - LsNull	No line.

NOTE: Using a value different from 0 (**LsSolid**) in this property, together with the **Width** property greater than 1 (one), may degrade the drawing performance of the Pen.

4.6.4.3.2.26 PenType

Determines the drawing type of the Pen in the E3Chart:

- **0**: Line
- **1**: Point
- **2**: Point-line
- **3**: Area

Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.PenType = 1
End Sub
```

4.6.4.3.2.27 QueryName

Determines the name of the Query the Pen is using. This property is used if the **DataSourceType** property is set to 1 (Historic). Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.QueryName = Query12
End Sub
```

4.6.4.3.2.28 ScaleX

The **ScaleX** property indicates to which X scale of the E3Chart this Pen is linked. The configured scale for **ScaleX** has a horizontal orientation, that is, it may be positioned on top or on the bottom of the E3Chart. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.AddPen("DemoTagPen2")
  Pen.XLink = "Data.DemoTag2"
  Pen.UseTimeStamp = True
  ' The scale must exist.
  Pen.ScaleX = "ScaleForDemoTag2"
  Pen.Connect
End Sub
```

4.6.4.3.2.29 ScaleY

9 The **ScaleY** property indicates to which Y scale of the E3Chart this Pen is linked. The configured scale for **ScaleY** has a vertical orientation, positioned on the left or on the right of the object. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.AddPen("DemoTagPen2")
  Pen.YLink = "Data.DemoTag2"
  Pen.UseTimeStamp = True
  ' The scale must exist.
  Pen.ScaleY = "ScaleForDemoTag2"
  Pen.Connect
End Sub
```

4.6.4.3.2.30 ScanValue

9 Defines the scan time waiting time for the real time Pen tag. This value is taken into account on analog drawing mode. When this value overrides the value determined by **ScanValue**, it is considered that the tag value has not been changed on the interval. Otherwise, when **ScanValue** is equal to zero, Pen data is always connected with a straight line bounding these two points as if the value is varying in linear mode. The unit of this property is milliseconds. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.ScanValue = 1000
End Sub
```

4.6.4.3.2.31 ShowAverage

◩ Enables displaying Pen average on the E3Chart. This property is only effective if the **EnableCalc** property is enabled.

4.6.4.3.2.32 ShowBadPoints

◩ When disabled, bad quality points are not plotted. If the Pen is drawing lines, the lines crossing bad quality points are not connected. For the point quality to be considered in the historic part of the Pen, the **fieldname_quality** must be selected on the E3Chart Query. When enabled, all points are plotted normally.

4.6.4.3.2.33 ShowMinMax

◩ Enables displaying minimum and maximum points of the Pen on the E3Chart. This property is only effective if the **EnableCalc** property is enabled.

4.6.4.3.2.34 UseTimeStamp

Determines that it is used, for the horizontal Axis, the value of the *timestamp* associated to the vertical Axis. See an example in the description of the **DataSourceType** property.

4.6.4.3.2.35 Visible

Determines if the Pen is visible on the E3Chart. If this option is set to True, the Pen is visible at run time. Otherwise, the Pen is hidden. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Visible = Not Pen1.Visible
End Sub
```

4.6.4.3.2.36 Width

Determines the width of the Pen line on the E3Chart. The default value of this property is 0 (zero).

NOTE: Using values greater than 1 (one) in this property, together with the **PenStyle** property different from 0 (**LsSolid**), may degrade the drawing performance of the Pen.

4.6.4.3.2.37 XField

A Name of the Query Field used to plot data on a horizontal scale. It is used for historic Pens.

4.6.4.3.2.38 XLink

A Name of the *link* used to plot data on a horizontal scale. When the value of this property is changed, the Pen is automatically disconnected. After configuring it, the **Connect** method must be called so that the Pen starts receiving data relative to this *link*. This is only used for real time Pens.

4.6.4.3.2.39 XMaxY

Informs the X value relative to the **MaxY** point. This is a read-only property.

4.6.4.3.2.40 XMinY

Informs the X value relative to the **MinY** point. This is a read-only property.

4.6.4.3.2.41 YField

A Name of the Query Field used to plot data on a vertical scale. It is used for historic Pens.

4.6.4.3.2.42 YLink

A Name of the *link* used to plot data on a vertical scale. When the value of this property is changed, the Pen is automatically disconnected. After configuring it, the **Connect** method must be called so that the Pen starts receiving data relative to this *link*. This is used only for real time Pens.

4.6.5 Axis Collection

This section contains information about **methods** and **properties** of the Axis Collection object. This object does not have events associated to it.

4.6.5.1 Methods

This section contains information about the methods of the Axis Collection object.

4.6.5.1.1 AddAxis

AddAxis(*AxisName*)

Adds a new Axis with the name specified in the *AxisName* parameter and returns it. In case of trying to create an Axis with an already existing name, an error message is displayed. To generate a name automatically, simply pass a blank name in the *AxisName* parameter. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set newAxis = Chart.Axes.AddAxis("")
  newAxis.Color = RGB(255, 0, 0)
End Sub
```

4.6.5.1.2 Remove

Remove(*Index*)

Removes the Axis by its name or index, as specified in the *Index* parameter. Main Axes 0 and 1 cannot be removed. In case of trying to remove them, an error message is displayed. Example:

```
Sub CommandButton1_Click()
  ' This example removes all additional axes
  Set Chart = Screen.Item("E3Chart")
  While (Chart.Axes.Count > 2)
    Chart.Axes.Remove(2)
  Wend
```

```
End Sub

Sub CommandButton1_Click()
  ' Removes an additional axis, if available
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Remove(2)
End Sub
```

4.6.5.2 Properties

This section contains information about the properties of the Axis Collection object.

4.6.5.2.1 Count

Returns the total amount of E3Chart Axes, including the main Axes (Horizontal and Vertical). Example:

```
CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  MsgBox Chart.axes.Count
End Sub
```

4.6.5.2.2 HorAxis

Returns the main horizontal Axis. This Axis also appears on the Axes list. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  MsgBox Chart.axes.Item(0).Name & ", " & _
    Chart.axes.Item(1).Name
  MsgBox Chart.axes.HorAxis.Name & ", " & _
    Chart.axes.Item("AxisName").Name
End Sub
```

4.6.5.2.3 Item

Returns the Axis using its name or index. The index 0 (zero) is always the main horizontal Axis, and the index 1 is always the main vertical Axis. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart")
  MsgBox Chart.axes.Item(0).Name & ", " & _
    Chart.axes.Item(1).Name
  MsgBox Chart.axes.HorAxis.Name & ", " & _
    Chart.axes.Item("AxisName").Name
End Sub
```

4.6.5.2.4 VerAxis

Returns the main vertical Axis. This Axis also appears on the list of Axes. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
```

```
  MsgBox Chart.axes.Item(0).Name & ", " & _
    Chart.axes.Item(1).Name
  MsgBox Chart.axes.Item("AxisName").Name & ", " & _
    Chart.axes.VerAxis.Name
End Sub
```

4.6.5.3 Axes

This section contains information about **methods** and **properties** of the Axis object. This object does not have events associated to it.

4.6.5.3.1 Methods

This section contains information about the methods of the Axis object.

NOTE: The **HorAxis** and **VerAxis** properties are from the Axes Collection, and access default vertical and horizontal Axes, respectively. For example, instead of using "Chart.Axes.Item('HorizontalAxis')", users can use "Chart.Axes.HorAxis". Other user-created Axes have their own names.

4.6.5.3.1.1 GetHistoricPeriod

GetHistoricPeriod(*Begin*, *End*)

Returns the time interval shown in the historic scale. The *Begin* parameter indicates the historic scale's initial date and the *End* parameter indicates the final date. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart")
  Chart.Axes.Item("AxisName").GetHistoricPeriod min, max
  Value = CStr(dmin) & " " & CStr(dmax)
  MsgBox "Start Date = " & CStr(min) & _
    vbNewLine & " Ending Date = " & CStr(max)
End Sub
```

4.6.5.3.1.2 GetMinMax

GetMinMax(*Min*, *Max*)

Returns the minimum and maximum values of the numeric scale in the *Min* and *Max* parameters, respectively. Example:

```
Sub CommandButton1_DBClick()
  Set Chart = Screen.Item("E3Chart")
  Chart.Axes.Item("AxisName").GetMinMax dmin, dmax
  MsgBox CStr(dmin) & " " & CStr(max)
End Sub
```

4.6.5.3.1.3 GetRealTimePeriod

GetRealTimePeriod(*Period*)

Returns the time unit configured in the real-time scale. The *Period* parameter receives the value of the time scale. Time units available are described on the next table.

Available time units

VALUE	DESCRIPTION
0 - tuSeconds	Time unit in seconds
1 - tuMinutes	Time unit in minutes
2 - tuHours	Time unit in hours
3 - tuDays	Time unit in days
4 - tuWeeks	Time unit in weeks
5 - tuMonths	Time unit in months
6 - tuYears	Time unit in years

Example:

```
Dim Unit, Value
Unit = Screen.Item("E3Chart1").Axes.Item_
  ("HorizontalAxis").GetRealTimePeriod(Value)
MsgBox "Value: " & CStr(Value) & " Unit: " & CStr(Unit)
```

4.6.5.3.1.4 GetTickSpacing

GetTickSpacing(*TickSpacing*, *TimeUnit*)

Returns the spacing between ticks (scale subdivisions) and the configured unit. The *TickSpacing* parameter determines the spacing between ticks and the *TimeUnit* parameter determines the unit. When this parameter is zero, this means that it is automatic. The unit is not used when scale is numeric. The available values for the *TimeUnit* parameter are described on the **GetRealTimePeriod** method. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Unitvalue_
    = Chart.Axes.Item("AxisName").GetTickSpacing(TickSpacing)
  MsgBox "Value " = " & CStr(TickSpacing) & _
  " unit " & CStr(Unitvalue)
End Sub
```

4.6.5.3.1.5 SetHistoricPeriod

SetHistoricPeriod(*Begin*, *End*)

Configures a time period for the historic scale. The *Begin* parameter determines the scale's initial period and the *End* parameter determines the scale's final period.

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart2")
  Chart.Axes.Item("AxisName").ScaleType = 2
  ' Displays the last period of time
  Chart.Axes.Item("AxisName").SetHistoricPeriod now - 1, now
End Sub
```

4.6.5.3.1.6 SetMinMax

SetMinMax(*Min*, *Max*)

Configures minimum and maximum values for the numeric scale. The minimum value is determined by the *Min* parameter and the maximum value by the *Max* parameter. Example:

```
Sub Circle1_Click()
  Set Chart = Screen.Item("E3Chart2")
  Chart.Axes.Item("AxisName").SetMinMax -10, 500
End Sub
```

4.6.5.3.1.7 SetRealTimePeriod

SetRealTimePeriod(*Times*, *TimeUnit*)

Configures a time period in the unit defined by the *TimeUnit* parameter. The available options in this parameter are described on the **GetRealTimePeriod** method. The *Times* parameter determines the time interval and the scale unit is specified by the *TimeUnit* parameter. The Axis is always updated in this mode (real time). Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart5")
  ' tuSeconds = 0, tuMinutes =1, tuHours = 2, tuDays = 3,
  ' tuWeeks = 4, tuMonths = 5, tuYears = 6
  ' 2 minutes
  Chart.Axes.Item("AxisName").SetRealTimePeriod 2, 1
  Chart.Axes.Item("AxisName").SetTickSpacing 30, 0
End Sub
```

4.6.5.3.1.8 SetTickSpacing

SetTickSpacing(*TickSpacing*, *TimeUnit*)

Configures the spacing between ticks (scale subdivisions) by using the configured unit. The spacing between ticks is determined by the *TickSpacing* parameter. The *TimeUnit* parameter determines the unit. If the scale is numeric, this unit is not considered. The available options in the *TimeUnit* parameter are described on the **GetRealTimePeriod** method. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
```

```
  ' 10 (if scale is numeric,
  ' the value of the unit is not considered)
  Chart.Axes.Item("AxisName").SetTickSpacing 10, 0
  Chart.Axes.Item("AxisName").SetTickSpacing 20, 0
End Sub
```

4.6.5.3.2 Properties

This section contains information about the methods of the Axis object.

NOTE: The **HorAxis** and **VerAxis** properties are from the Axes Collection, and access default vertical and horizontal Axes, respectively. For example, instead of using "Chart.Axes.Item('HorizontalAxis')", users can use "Chart.Axes.HorAxis". Other user-created Axes have their own names.

4.6.5.3.2.1 Color

Determines the Axis' main color.

4.6.5.3.2.2 EnableTextColor

This property, when enabled, specifies that the Axis' text has the same color of the scale configured in the **Color** property. The default value of this property is False.

4.6.5.3.2.3 Format

Determines the format of Axis values. Formats allowed in this property are described on **E3 User's Manual**, on topic **Screens and Screen Objects - Value Formatting**, or leave it blank to select the **Automatic** mode. This property allows using the | (vertical bar) character on the formatting **String** as a line break.

If Axis formatting is selected as **Automatic** (Axis Properties window, **Scale** tab, **Formatting** group) and the type of scale is selected as **Show latest data (Real-Time)** or **Time interval (historical data)**, date and time format obeys regional configuration of the Windows user. If the scale type is selected as **Numeric scale**, then it uses an automatic formatting of numbers. Example:

```
Sub CommandButton1_Click()
  ' Changes formatting
  Set Chart = Screen.Item("E3Chart1")
  strOldFormat = Chart.Axes.Item("AxisName").Format
  MsgBox "Click to set automatic formatting."
  Chart.Axes.Item("AxisName").Format = "" ' Automatic
  MsgBox "Click to use another formatting."
  Chart.Axes.Item("AxisName").Format = "0.0"
  MsgBox "Click to use another formatting."
  Chart.Axes.Item("AxisName").Format = "dd/MM/yy hh:mm:ss"
  MsgBox "Click again to change back
    to the original formatting."
```

```
  Chart.Axes.Item("AxisName").Format = strOldFormat
End Sub
```

4.6.5.3.2.4 GridColor

Determines the color of grid lines. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").ShowGrid = False
  MsgBox "Click to change the color of the grid lines."
  Chart.Axes.Item("AxisName").GridColor = RGB(0, 0, 255)
  Chart.Axes.Item("AxisName").ShowGrid = True
End Sub
```

4.6.5.3.2.5 GridStyle

Determines the grid line's style. The available options are described on the next table.

Available options for GridStyle

OPTION	DESCRIPTION
0 - solid	The grid line's style is solid
1 - dash	The grid line's style is dashed
2 - dot	The grid line's style is dotted
3 - dashdot	The grid line's style is dash-dot
4 - dashdotdot	The grid line's style is dash-dot-dot
5 - invisible	There is no visible lines on the grid

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  ' Solid 0, dash 1, dot 2, dash-dot 3,
  ' dash-dot-dot 4, invisible 5
  For i = 0 To 5
    MsgBox "Click to change the line style of the grid."
    Chart.Axes.Item("AxisName").GridStyle = i
  Next
End Sub
```

4.6.5.3.2.6 Inverse

Inverts the order of the minimum and maximum values on the numerical scale. Normally, in vertical scales, the minimum value appears at the bottom and the maximum at the top. On horizontal scales, the minimum value appears on the left and the maximum value on the right. When the **Inverse** property is set to True, however, this order is inverted, maximum values at the bottom or on the left and minimum values at the top or on the right. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").Inverse = Not _
    Chart.Axes.Item("AxisName").Inverse
End Sub
```

4.6.5.3.2.7 MinorTicks

9 Determines the total amount of subdivisions among scales. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").MinorTicks = _
    Chart.Axes.Item("AxisName").MinorTicks + 1
End Sub
```

4.6.5.3.2.8 Mirror

◩ Indicates the Axis mirroring. If this property is set to True, the Axis is mirrored on the opposite side of the original Axis. Otherwise, the Axis remains on the same position. Example:

```
Sub CommandButton1_DBClick()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").Mirror = Not _
    Chart.Axes.Item("AxisName").Mirror
End Sub
```

4.6.5.3.2.9 Name

A Determines the Axis' name.

4.6.5.3.2.10 Position

Determines the Axis position relative to E3Chart's grid. The available options are described on the next table.

Available options for Position

OPTION	DESCRIPTION
0 - axpLeft	Axis is placed on scale's left side
1 - axpRight	Axis is placed on scale's right side
2 - axpTop	Axis is placed at scale's top
3 - axpBottom	Axis is placed at scale's bottom

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set newAxis = Chart.Axes.AddAxis("")
  For i = 0 To 3
```

```
    MsgBox "Click OK to change the axis position."
    newAxis.Position = i
  Next
  MsgBox "Remove the axis."
  Chart.Axes.Remove(newAxis.Name)
End Sub
```

4.6.5.3.2.11 ScaleType

Determines the type of scale displayed by the Axis. The available options for this item are described on the next table.

Available options for ScaleType

OPTION	DESCRIÇÃO
0 - atNumberScale	Numerical scale
1 - atLastPeriod	Displays the last period (Real Time)
2 - atPeriod	Time interval (Historic)

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set newAxis = Chart.Axes.AddAxis("")
  For i = 0 To 2
    MsgBox "Click OK to change the scale type."
    newAxis.ScaleType = i
  Next
  MsgBox "Remove the axis."
  Chart.Axes.Remove(newAxis.Name)
End Sub
```

4.6.5.3.2.12 ShowGrid

Determines if grid lines are visible. If this property is set to True, the grid lines are displayed. Otherwise, grid lines are hidden. Example:

```
Sub CommandButton_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").ShowGrid = Not _
  Chart.Axes.Item("AxisName").ShowGrid
End Sub
```

4.6.5.3.2.13 Title

Determines the Axis' title. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").Title = _
    Chart.Axes.Item("AxisName").Name
  MsgBox "Click to remove the title."
```

```
  ' Removes the title
  Chart.Axes.Item("AxisName").Title = ""
End Sub
```

4.6.5.3.2.14 Visible

◩ Determines the Axis' visibility on the grid. If this property is set to True, the Axis is visible on the grid. Otherwise, the Axis is invisible. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item(E3Chart1)
  Chart.Axes.Item("AxisName").Visible = Not _
  Chart.Axes.Item("AxisName").Visible
End Sub
```

4.6.6 Query Collection

This section contains information about **methods** and **properties** of the Query Collection object. This object does not have events associated to it.

4.6.6.1 Methods

This section contains information about the methods of the Query Collection object.

4.6.6.1.1 AddQuery

AddQuery(*QueryName*, *IsInternal*)

Adds a Query to the E3Chart Queries Collection. The *QueryName* parameter determines the Query's name. The *IsInternal* parameter is optional, deprecated, and must not be informed. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").Queries.AddQuery("Query1")
End Sub
```

4.6.6.1.2 Item

Item(*Index*)

Sets up or returns a Query Collection by name or by index. This method has an *Index* parameter that is mandatory, and assumes the following behavior: if it is a number, it corresponds to the Query's index in the Queries Collection. If it is a text, it corresponds to the Query's name. Example:

```
Sub CommandButton1_Click()
  Set query = Screen.Item("E3Chart1").Queries.Item(0)
End Sub
```

4.6.6.1.3 Remove

Remove(*Index*)

Removes the Query object, specified by name or by index. This method has an *Index* parameter corresponding to the Query's name to be removed. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").Queries.Remove(0)
End Sub
```

4.6.6.1.4 UpdateData

UpdateData()

Updates data in all Queries. Example:

```
Sub Text1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.Queries.UpdateData()
End Sub
```

4.6.6.2 Properties

This section contains information about the properties of the Query Collection object.

4.6.6.2.1 Count

9 Contains the total amount of E3Chart Queries. This is a read-only property.

4.6.7 Legend

This section contains information about **methods** and **properties** of the Legend object. This object does not have events associated to it.

4.6.7.1 Methods

In a Legend, many columns can be selected. Each column shows an information type, and has a corresponding name and value. The following table displays all possible columns in a Legend.

Available options to identify a Legend column

OPTION	VALUE	NAME	DESCRIPTION
AverageY	**10**	**AverageY**	Shows Pen's average value in this interval.
BeginX	**13**	**XBegin**	Shows cursor's initial position.

OPTION	VALUE	NAME	DESCRIPTION
BeginY	**17**	**YBegin**	Shows the interpolated point where initial cursor meets a Pen.
DiffX	**15**	**DiffX**	Shows the difference between initial and final cursors.
DiffY	**16**	**DiffY**	Shows the difference between initial and final interpolated points on the Y Axis.
EndX	**14**	**XEnd**	Shows cursor's final position.
EndY	**18**	**YEnd**	Shows the interpolated point where final cursor meets a Pen.
MaximumY	**12**	**MaxY**	Shows Pen's maximum value in this interval.
MinimumY	**11**	**MinY**	Shows Pen's minimum value in this interval.
Pen color	**6**	**Color**	Shows Pen's color.
Pen description	**5**	**Description**	Shows the text in Pen's **DocString** property.
Pen name	**0**	**Name**	Shows Pen's name.
Status	**7**	**Status**	Shows current Pen state.
Unit	**19**	**EU**	Shows Pen's engineering unit.
X tag name	**1**	**TagX**	Shows Tag's name linked to the X Axis.
X tag value	**3**	**TagXValue**	Shows search value on X Axis.
XScale	**8**	**ScaleX**	Shows X Axis' name linked to the Pen.
Y tag name	**2**	**TagY**	Shows Tag's name linked to the Y Axis.
Y tag value	**4**	**TagYValue**	Shows search value on Y Axis.
YScale	**9**	**ScaleY**	Shows Y Axis' name linked to the Pen.

4.6.7.1.1 ChangeColumnPos

ChangeColumnPos(*Source*, *Dest*)

Swaps two columns. This method has the following parameters:

- **Source**: is the index of the column to swap to *Dest*
- **Dest**: is the index of the column to swap to *Source*

Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").Legend.ChangeColumnPos 1, 2
End Sub
```

4.6.7.1.2 Count

Count()

Returns the number of columns of a Legend. Example:

```
Sub CommandButton1_Click()
  MsgBox Screen.Item("E3Chart1").Legend.Count()
End Sub
```

4.6.7.1.3 InsertColumn

InsertColumn(*Col*, *Index*)

Inserts a new column in a Legend. This method has the following parameters:

- **Col**: Identifies the column to insert (see the **Available options for column identification** table in the topic **Legend Methods**)
- **Index**: Determines the position where the column will be inserted

Example:

```
Sub CommandButton1_Click()
  'Displays the Pen's name
  Screen.Item("E3Chart1").Legend.InsertColumn 0, 0
End Sub

Sub CommandButton1_Click()
  'Displays the Pen's color
  Screen.Item("E3Chart1").Legend.InsertColumn "Color", 0
End Sub
```

4.6.7.1.4 Item

Item(*Col*)

Returns a Legend's column by name or by index. The *Col* parameter determines the column's index or name (see the **Available options for column identification** table in the **Legend Methods** topic). Example:

```
Sub CommandButton1_Click()
  MsgBox Screen.Item("E3Chart1").Legend.Item(1)
End Sub
```

4.6.7.1.5 RemoveColumn

RemoveColumn(*Col*)

Removes a column. This method has a *Col* parameter determining the column to remove (see the **Available options for column identification** table in the **Legend Methods** topic). Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").Legend.RemoveColumn(1)
End Sub
```

4.6.7.2 Properties

This section contains information about the properties of the Legend object.

4.6.7.2.1 BackColor

Configures or returns the Legend's background color. The default value of this property is white (**RGB(255, 255, 255)**).

4.6.7.2.2 EnableTextColor

This property, when enabled, specifies that the Legend's text has the same color of the Pen, configured in its **Color** property. The default value of this property is False.

4.6.7.2.3 LegendPos

Indicates the Legend's position on an E3Chart.

Available options for LegendPos

OPTION	DESCRIPTION
0 - lgTop	Displays the Legend on top.
1 - lgLeft	Displays the Legend on the left.
2 - lgBottom	Displays the Legend at the bottom.
3 - lgRight	Displays the Legend on the right.

Example:

```
Sub CommandButton10_Click()
  Screen.Item("E3Chart1").Legend.LegendPos = 3
End Sub
```

4.6.7.2.4 ShowAllPens

◩ When this property is set to True, all EChart's Pens are displayed on the Legend. The Pen's **Visible** property is ignored. When set to False, only Pens with their **Visible** property set to True are displayed. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.Legend.ShowAllPens = Not _
    E3Chart1.Legend.ShowAllPens
End Sub
```

4.6.7.2.5 ShowHeader

◩ Determines the visibility of a Legend's title (header). If this option is set to True, the Legend's title is displayed. Otherwise, the title is hidden. Example:

```
Sub CommandButton2_Click()
  Screen.Item("E3Chart1").Legend.ShowHeader = False
End Sub
```

4.6.7.2.6 Size

9 Determines the Legend's size. This size may be the height or width, depending on the Legend's position. Example:

```
Sub CommandButton13_Click()
  MsgBox Screen.Item("E3Chart1").Legend.Size
End Sub
```

4.6.7.2.7 Visible

◩ Determines the Legend's visibility. If this property is set to True, the Legend is visible on the E3Chart. Otherwise, it is hidden. Example:

```
Sub CommandButton13_Click()
  Screen.Item("E3Chart1").Legend.Visible = False
End Sub
```

4.6.7.3 Legend Columns

This section contains information about **properties** of the Legend Column object. This object does not have events nor methods associated to it.

4.6.7.3.1 Properties

This section contains information about the properties of the Legend Column object.

4.6.7.3.1.1 Caption

A Defines a title for the column.

4.6.7.3.1.2 Column

Returns a column identifier. See the **Available options for Columns** table.
Example:

```
Sub CommandButton15_Click()
  Set col= Screen.Item("E3Chart1").Legend.Item(1)
  MsgBox col.Column
End Sub
```

4.6.7.3.1.3 Format

A Configures a display format for the column. Example:

```
Sub CommandButton15_Click()
  Set col= Screen.Item("E3Chart1").Legend.Item(1)
  MsgBox col.Format = "0.0"
End Sub
```

4.6.7.3.1.4 Index

9 Returns the column position on the Legend. Example:

```
Sub CommandButton15_Click()
  Set col = Screen.Item("E3Chart1").Legend.Item(1)
  MsgBox col.Index
End Sub
```

4.6.7.3.1.5 Name

A Determines the column's name. Column names can be seen on the **Available options for columns** table. Example:

```
Sub CommandButton1_Click()
  set Chart = screen.item("E3Chart1")
  set query = Chart.Queries.Item(0)
  MsgBox query.Name
  MsgBox query.TableName
End Sub
```

4.6.7.3.1.6 TextAlign

Returns the column's text alignment. The available options for this property are the following:

Available options for TextAlign

OPTION	DESCRIPTION
0 - taLeft	Left alignment.
1 - taRight	Right alignment.
2 - taCenter	Centered alignment.

Example:

```
Sub CommandButton1_Click()
  Set col = Screen.Item("E3Chart1").Legend.Item(1)
  col.TextAlign = 2
End Sub
```

4.6.7.3.1.7 Width

Returns the column's width. Example:

```
Sub CommandButton1_Click()
  Set col = Screen.Item("E3Chart1").Legend.Item(1)
  col.width = 50
End Sub
```

4.7 E3Playback

This section contains information about **methods** and **properties** of the E3Playback object. This object does not have events associated to it.

4.7.1 Methods

This section contains information about the methods of the E3Playback object.

4.7.1.1 Pause

Pause()

Pauses the playback clock at the current time.

4.7.1.2 Play

Play()

Starts playing data at the current playback time, by moving the clock forward based on the current playing speed.

4.7.1.3 Stop

Stop()

Interrupts playback, Tag data and alarms are removed from the Screen, and no new query is performed on configured database.

4.7.2 Properties

This section contains information about the properties of the E3Playback object.

4.7.2.1 CurrentTime

Displays current date and time of this E3Playback object. This property is only available at run time.

4.7.2.2 DBServer

Indicates the name of the Database that contains application's historical data.

4.7.2.3 InitialScreen

Indicates the Screen that is initially displayed on E3Playback. If this property is left blank, then it uses the Screen or Frame configured in Viewer's **InitialScreen** property.

4.7.2.4 PlaybackState

Informs the status of this E3Playback object. This property is only available at run time and its options are described on the next table.

Options for PlaybackState property

OPTION	DESCRIPTION
0 - Stopped	Playback clock is stopped.
1 - Playing	Playback clock is moving.
2 - Paused	Playback clock is paused.

4.8 Reports

This section contains information about **events**, **methods**, and **properties** of the Report object.

4.8.1 Events

This section contains information about the events of the Report object.

4.8.1.1 OnAfterPrint

OnAfterPrint()

Started after a Section has been assembled in the Report. Users can use this event to update any necessary counter after finishing the Report.

4.8.1.2 OnBeforePrint

OnBeforePrint()

Started before the Section has been assembled in the Report. Users can use this event to modify an object's value in the Report before printing it. It is recommended that Report's Query fields do not be accessed while this event is being used.

4.8.1.3 OnDataInitialize

OnDataInitialize()

Occurs before the **OnReportStart** event. This event allows adding and configuring fields to the Report's **Fields** collection, before generating them. Example:

```
Sub OnDataInitialize()
  Fields.Add "Name"
  Fields.Add "Sector"
  Fields.Add "Code"
End Sub
```

4.8.1.4 OnError

OnError(*Number*, *Description*, *SCode*, *Source*, *HelpFile*, *HelpContext*, *CancelDisplay*)

Generated by a Report's internal error. If this event is not handled, E3 then displays a generic error message.

Parameters of an OnError event

NAME	DESCRIPTION
Number	Integer identifying the error.
Description	**String** with the error description.
SCode	Integer with the error code of the OLE subsystem (not used).
Source	**String** with the object that generated the error.
HelpFile	**String** with the name and path of the help file.
HelpContext	Number of the help topic's context referring to the error (integer).
CancelDisplay	**Boolean** indicating whether the error should be displayed on a **MessageBox**.

4.8.1.5 OnFetchData

OnFetchData(*eof*)

Triggered every time a new record is processed. This event is used to run a script that modifies field values added to a Report, in a script linked to the **OnDataInitialize** event. The default value of the *eof* parameter is True, and indicates that after the script the processing of the Report's current record is finished.

4.8.1.6 OnFormat

OnFormat()

Started before data is read and loaded in the Report, but before the Section is prepared for printing. This event can be used to modify the layout of the Report Section, or any other object.

4.8.1.7 OnHyperlink

OnHyperlink(*Button*, *Link*)

Occurs when clicking a link on a Report. Users can use this event to run a script that redirects a link, or to configure a link on the Report. The *Button* parameter indicates what button was clicked (normally, 1) and the *Link* parameter determines what is the address to follow.

4.8.1.8 OnNoData

OnNoData()

Occures when there is no data to print on a Report. Users can use this event to run a script that displays an error message on the screen, warning that there is no data to print and then cancel Report printing.

4.8.1.9 OnPageEnd

OnPageEnd()

Occurs at the end of each Report's page printing.

4.8.1.10 OnPageStart

OnPageStart()

Occurs at the beginning of each Report's page printing.

4.8.1.11 OnPrintProgress

OnPrintProgress(*PageNumber*)

Occurs while a Report page is being printed. The *PageNumber* parameter indicates the current page number.

4.8.1.12 OnReportEnd

OnReportEnd()

Triggered at the end of the Report generation, after printing it.

4.8.1.13 OnReportStart

OnReportStart()

Triggered at the beginning of the Report generation, before start printing it.

4.8.2 Methods

This section contains information about the methods of the Report object.

4.8.2.1 Export

Export([*ExportFilter*[, *ExportFileName*]])

Prints a Report, according to the specified format. This method has an *ExportFilter* parameter, which determines the Report filter, indicating the export format. It can assume the following options:

- **PDF**: Exports data to the Adobe PDF format
- **Excel**: Exports data to the Excel spreadsheet format
- **HTML**: Exports data to the HTML format
- **TEXT**: Exports data to a text file
- **RTF**: Exports data to the Rich Text Format
- **TIFF**: Exports data to the Tag Image File Format

When simply informing a filter name, as previously exposed, data is exported using common filter properties. Users can modify the common properties of export filters by using the **GetExportFilter** method before exporting data. The name of the file must be informed in the *ExportFileName* parameter. Example:

```
Sub Button1_Click()
  Set report = Application.LoadReport("[Report3]")
  Select case Application._
```

```
    SelectMenu("PDF|Excel|HTML|RTF|Text|TIFF|Text(CSV)")
    Case 1
      Report.Export "PDF", "C:\mail\reports\report.pdf"
      MsgBox "Exported to PDF!"
    Case 2
      Report.Export "EXCEL", "C:\mail\reports\report.XLS"
      MsgBox "Exported to XLS!"
    Case 3
      Report.Export "HTML", "C:\mail\reports\report.html"
      MsgBox "Exported to HTML!"
    Case 4
      Report.Export "RTF", "C:\mail\reports\report.rtf"
      MsgBox "Exported to RTF!"
    Case 5
      Report.Export "TEXT", "C:\mail\reports\report.txt"
      MsgBox "Exported to TXT!"
    Case 6
      Report.Export "TIFF", "C:\mail\reports\report.tiff"
      MsgBox "Exported to TIFF!"
    Case 7
      Set reportFilter = report.GetExportFilter("TEXT")
      reportFilter.FileName = "C:\mail\reports\report2.txt"
      reportFilter.TextDelimiter = ","
      report.Export reportFilter
      MsgBox "Exported to TXT using a filter!"
  End Select
End Sub
```

4.8.2.2 GetExportFilter

GetExportFilter(*FilterName*)

Returns an object that specifies custom export parameters. This method has the *FilterName* parameter, which determines the Report filter, indicating the format type for export. It can assume the following options:

- **PDF**: Exports data to the Adobe PDF format
- **Excel**: Exports data to the Excel spreadsheet format
- **HTML**: Exports data to the HTML format
- **TEXT**: Exports data to a text file
- **RTF**: Exports data to the Rich Text Format
- **TIFF**: Exports data to the Tag Image File Format

After getting the filter, the following properties can be modified:

Properties that can be changed

PROPERTY	FILTER	DESCRIPTION
AutoRowHeight	Excel	In True (default), automatically configures the row height. In False,configures the size to the largest element on the row.
BorderSpace	Excel	Minimum space between cells. The default value is 59 twips.
CreateCSSFile	HTML	If True, generates a CSS file in the *HTMLOutputPath* directory.
DoubleBoundaries	Excel	In True, indicates that elements aligned to the right must replace the ones aligned to the left on the same column. Otherwise, leave it in False to make more room.
ExportRange	HTML	Indicates a page range for export. For example, "1, 2, 3-9, 14".
FaxExport	TIFF	Object that allows exporting data in RFC 1314 TIFF format.
FileName	All	Informs the file name to export current data.
GenPageBreaks	HTML	In True, places page breaks under the lowest element in each Report's page.
HTMLOutputPath	HTML	Default HTML file output path.
JPGQuality	PDF	Indicates the quality level of the exported images (from 0 to 100).
MinColumnWidth	Excel	Minimum size of the column. The default value is 1011 twips.
MinRowHeight	Excel	Minimum size of the row.
MultiSheet	Excel	In True, each Report page goes to a separate spreadsheet.
PageDelimiter	Text	Configures or returns the delimiter character between pages.
SuppressEmptyLines	Text	Removes or inserts empty lines, for layout purposes.

PROPERTY	FILTER	DESCRIPTION
TextDelimiter	HTML	Configures or returns the delimiter character among texts.
TrimEmptySpace	Excel	In True, the vertical space among elements is removed. The default is False.
Unicode	Text	Determines whether the text is saved in Unicode format (16 bits).
WebCacheOutput	HTML	In True, the Report is exported to the WebCache service. Otherwise (default), it is not exported.

4.8.2.3 Print

Print()

Prints a Report. Example:

```
Sub Rect_Click()
  Application.LoadReport("[Report3]").Print()
End Sub
```

4.8.2.4 PrintPreview

PrintPreview()

Performs an on-screen preview of the Report. If the Report is correctly displayed on screen, returns True. In case the user clicks **Cancel**, or any error occurs, returns False. Example:

```
Sub CommandButton1_Click()
  Set report = Application.LoadReport("[Report1]")
  Start = Application.GetObject("Data.Chart.datas").Value
  End = Application.GetObject("Data.Chart.datae").Value
  report.Item("Query1").SetVariableValue "Begin", Start
  report.Item("Query1").SetVariableValue "End", End
  report.PrintPreview()
End Sub
```

NOTE: This method is not available for Reports loaded using the Server's **LoadReport** method.

4.8.2.5 Query

Query()

Returns the Query object currently selected on the Report. For more information

about this object, see the **Query** chapter. Example:

```
Sub Rect_Click()
  Set Query = Application.LoadReport("[Report3]").Query()
  Query.SetVariableValue("Key1", "XYZ")
End Sub
```

4.8.3 Properties

To create a script on a Report, use the Report Script Editor, available by clicking **Scripts Editor**. To view the Report, click **Generate Report**, both buttons are located on the **Report** tool bar.

Report scripts use some procedures, depending on the object or the section where the code can be inserted. For example:

```
Report.Sections("PageHeader").Controls("E3Chart1")._
  GridBkColor= RGB(255, 0, 255).
```

Where:

- **PageHeader**: is the name of the Section where the object is inserted on the Report
- **E3Chart1**: is the name of the object that is inside the specified Section, in this case it is PageHeader
- **GridBkColor**: is the object's property name, in this case it is E3Chart
- **RGB (255, 0, 255)**: it is the property's parameter. In this case, it changes the chart's background color to pink

Thus, to create a script on a Report, use the following concept:

```
Report.Sections("SectionName").Controls("ObjectName")._
  PropertyName = property_parameters
```

NOTE: The Report object encapsulates an ActiveReport object (or AR), which is the Report itself.

4.8.3.1 Caption

A Contains the Report's title that appears in the title bar of the *preview* window. The default value of this property is an empty **String**.

4.8.4 Layout

This section contains information about **properties** of the Report's Layout object. This object does not have events nor methods associated to it.

4.8.4.1 Properties

This section contains information about the properties of the Report's Layout object.

NOTE: The properties described here are part of the ActiveReport (AR) object, which is encapsulated on the Report. These properties are only valid inside the AR scope, and cannot be accessed from outside this object.

4.8.4.1.1 _PageBottomMargin

9 Determines the Report's bottom margin, in twips (one twip is equal to 1/440 inches). The default value of this property is 1440 (one inch or 2,54 cm).

4.8.4.1.2 _PageLeftMargin

9 Determines the Report's left margin, in twips (one twip is equal to 1/440 inches). The default value of this property is 1440 (one inch or 2,54 cm).

4.8.4.1.3 _PageRightMargin

9 Determines the Report's right margin, in twips (one twip is equal to 1/440 inches). The default value of this property is 1440 (one inch or 2,54 cm).

4.8.4.1.4 _PageTopMargin

9 Determines the Report's top margin, in twips (one twip is equal to 1/440 inches). The default value of this property is 1440 (one inch or 2,54 cm).

4.8.4.1.5 AllowSplitters

◩ Allows the Report's viewing area to be separated into two parts. This property is only available at run time. If this property is set to False (default value), the split bar does not appear on the screen.

4.8.4.1.6 documentName

A Determines the document name for the Report. This name appears on the Print Manager, and can be used to identify the Report. The default value of this property is "ActiveReports Document".

4.8.4.1.7 MaxPages

9 Establishes the maximum amount of pages for the Report. When this number is reached, E3 stops processing the document. The default value of this property is 10.

4.8.4.1.8 ParentReport

A This property is an internal system variable, and contains a reference to the Report object. This is a read-only property, and valid only for the **OnDataInitialize** and **OnReportEnd** events.

4.8.4.1.9 PrintWidth

9 Determines the width of the Report's print area, in twips. If the Report's size changes at run time, the printing width must also be changed, to ensure that the Report uses the whole print area. The size of the print area must also include the margin width, so that the Report do not oversize the paper size. If this happens, the error appears as a red-dotted line printed on every Report page.

4.8.4.1.10 RulerVisible

◩ When set to True, indicates that one vertical and one horizontal ruler are displayed on the Report's preview window. Otherwise, these rulers are invisible.

4.8.4.1.11 ScriptDebuggerEnabled

◩ Enables or disables the ActiveReport debugger (JIT), to debug scripts linked to the Report. This debugger is not available in E3, only on Reports.

4.8.4.1.12 ScriptLanguage

A Indicates the programming language used to interpret scripts linked to a Report. The default language is "VBScript", but "JScript" may also be used.

4.8.4.1.13 ShowParameterUI

◩ Enables or disables the Query's dialog box parameters that appear when the Report is being executed. If this property is set to True, the Query's dialog box parameters are displayed. Otherwise, these parameters are not displayed.

4.8.4.1.14 Status

Returns the Report status. The available options for this property are the following:

Available options for Status

OPTION	DESCRIPTION
0 - DDStatIdle	Indicates that the Report is closed.
1 - DDStartRunning	Indicates that the Report is being executed.
2 - DDStartCompleted	Indicates that the Report is completed.

OPTION	DESCRIPTION
3 - DDStartCanceled	Indicates that the Report has been canceled.

4.8.4.1.15 TOCEnabled

Enables or disables the Report's table of contents. If this property is set to True, the Report's table of contents is enabled. Otherwise, the Report has no table of contents. The default value of this property is True.

4.8.4.1.16 TOCVisible

If this property is set to True, the Report's table of contents is displayed. Otherwise, the table of contents remains invisible. The default value of this property is True.

4.8.4.1.17 ToolbarVisible

Enables or disables the toolbar of the Report's print preview window. If this property is set to True, the toolbar is enabled. Otherwise, this window has no toolbar.

4.8.4.1.18 UserData

A Configures or returns specific user information. This property is similar to the Visual Basic's **Tag** property, but it is exported and saved to an .rpx file. It can be used to save and load any customized information needed to draw the Report.

4.8.4.1.19 Version

A Returns the Report's version number.

4.8.4.1.20 WaterMark

Adds a background image to the Report (a watermark). Watermarks are texts or images that appear below the document text. Generally, they make the document visually more interesting.

Example of a watermark

The default value of this property is empty (no image or text).

4.8.4.1.21 WaterMarkAlignment

Determines the watermark alignment in the Report. The available options for this property are the following:

Available options for WaterMarkAlignment

OPTION	DESCRIPTION
0 - ddPATopLeft	Aligns the image to the top and left.
1 - ddPATopRight	Aligns the image to the top and right.
2 - ddPACenter	Aligns the image to the center (default).
3 - ddPABottomLeft	Aligns the image to the bottom and left.
4 - ddPABottomRight	Aligns the image to the bottom and right.

4.8.4.1.22 WaterMarkPrintOnPages

Indicates the number of Report pages which receive the watermark. The used syntax may include a single page, a page range, or even a combination of both. For example: 1, 5-8, 9, 10-15.

4.8.4.1.23 WaterMarkSizeMode

Configures the watermark size effect in the Report page. The available options are the following:

Available options for WaterMarkSizeMode

OPTION	DESCRIPTION
0 - ddSMClicp	The watermark is displayed in its original size in the Report.
1 - ddSMStretch	The watermark fills in the whole Report page.
2 - ddSMZoom	The watermark is resized up to the page size of the Report.

4.8.5 Section

This section contains information about common **properties** of the Report's Section object. This object does not have events nor methods associated to it.

4.8.5.1 Common Properties

This section contains information about the common properties of the Report's Section object.

4.8.5.1.1 BackColor

Specifies the Report Section's background color. The effect of this property is only visible if the **BackStyle** property is set to **1 - ddBKNormal**. The default value is white (**RGB(255, 255, 255)**).

4.8.5.1.2 BackStyle

Specifies the Report Section's background style. The available options for this property are the following:

- **0 - ddBKTransparent**: transparent background
- **1 - ddBKNormal**: normal background

4.8.5.1.3 CanGrow

Determines the application of a *stretch* effect to the text of a Report's page. In case the page's width or height are enlarged, the text accompanies this variation. If this property is set to True, the text accompanies object's height and width variations. Otherwise, the object gets its initial settings. The default value of this property is True.

4.8.5.1.4 CanShrink

◩ Determines the application of a *shrink* effect to the text of a Report's page. In case the page's width or height are diminished, the text accompanies this variation. If this property is set to True, the text accompanies the object's height and width variations. Otherwise, the object gets its initial settings. The default value of this property is True.

4.8.5.1.5 height

9 Determines the Report Section's page height. The default value of this property is 360.

4.8.5.1.6 IsRepeating

◩ Determines a Section repetition on the last page of the Report. If this property is set to True, the Section is repeated on the last page. Otherwise, there is no repetition.

4.8.5.1.7 Name

A Indicates the Report's Section name.

4.8.5.1.8 Type

Returns the Section type. The available options are:

Available options for Type

OPTION	DESCRIPTION
0 - ReportHeader	Report Header Section type.
1 - ReportFooter	Report Footer Section type.
2 - PageHeader	Page Header Section type.
3 - PageFooter	Page Footer Section type.
4 - GroupHeader	Group Header Section type.
5 - GroupFooter	Group Footer Section type.
6 - Detail	Detail Section type.

4.8.5.1.9 Visible

◩ Enables or disables Report's Section visibility. If this property is set to True, the Section is visible on the Report. Otherwise, it is hidden. The default value of this property is True.

4.8.5.2 Group Header

This section contains information about **properties** of the Report's Group Header object. This object does not have events nor methods associated to it.

4.8.5.2.1 Properties

This section contains information about the properties of the Report's Group Header object.

4.8.5.2.1.1 ColumnLayout

Determines if the **GroupHeader** is using the same column layout configured on the **Detail** Section. If this property is set to True, the number of columns in the **Detail** Section is the same as the associated **GroupHeader** or **GroupFooter**. Otherwise, it remains with its standard configuration.

4.8.5.2.1.2 DataField

Returns data from Report fields. Defines a mandatory field for a group inside the content of the **Detail** Section. This value is set to the name of all fields in the Report's data source, or to the name of a custom field inserted in the Fields collection. When this property is set, the Report creates a new group every time the field value changes in the Detail Section data recordings.

4.8.5.2.1.3 GrpKeepTogether

Determines if the **GroupHeader** Section prints as a single block on the same Report's page. The available options are the following:

Available options for GrpKeepTogether

OPTION	DESCRIPTION
0 - GrpNone	The page can be broken immediately after a **GroupHeader**.
1 - GrpFirstDetails	The **GroupHeader** prints with the first **Detail** Section of the same Report page or column.
2 - GrpAll	The **GroupHeader**, **Detail**, and **Footer** of a group print together on the same Report page.

The default value of this property is **0 - GrpNone**.

4.8.5.2.1.4 KeepTogether

Determines whether the Report Sections print as a single block, on the same page. The available options for this property are the following:

Available options for KeepTogether

OPTION	DESCRIPTION
0 - ddGrpNone	There is a page break after the Report.
1 - ddGrpFirstDetail	The Report prints the **Detail** Section on the same page or column.

4.8.5.2.1.5 NewColumn

Inserts a new column break before or after printing the Report Section. The available options for this property are the following:

Available options for NewColumn

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break on the Section.
1 - ddNBefore	Starts printing the Section on a new page.
2 - ddNPAfter	Starts a new page after printing the Section.
3 - ddNPBeforeAfter	Starts printing on a new page, and a new page after printing the Section.

4.8.5.2.1.6 NewPage

Inserts a page break in the Report. The available options for this property are the following:

Available options for NewPage

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in the Section (default).
1 - ddNBefore	Starts printing on a new page.
2 - ddNPAfter	Starts printing a new page after printing the Section.
3 - ddNPBeforeAfter	Starts printing on a new page, and starts a new page after printing the Section.

4.8.5.2.1.7 Repeat

Determines whether the **GroupHeader** is printed again after being linked to the **Detail** Section, when there are multiple pages, columns, or page breaks in the Report. The available options are the following:

Available options for Repeat

OPTION	DESCRIPTION
0 - ddRepeatNone	There is no re-printing of the Group Header (default).
1 - ddRepeatOnPage	Prints the header groups on top of the page, according to the specifications of the **Detail** Section.
2 - ddRepeatOnColumn	Prints the header groups on top of the Report's page column, according to the specifications of the **Detail** Section.
3 - ddRepeatAll	Prints the header groups and the other objects on top of the Report page, according to the specifications of the **Detail** Section.

4.8.5.2.1.8 UnderlayNext

Determines whether the Section must print one Section after another, consecutively. If this property is set to True, the next Section starts printing from the Section's upper coordinate on a Report page. Otherwise, this resource is not used. The default value of this property is False.

4.8.5.3 Detail

This section contains information about **properties** of the Report's Detail object. This object does not have events nor methods associated to it.

4.8.5.3.1 Properties

This section contains information about the properties of the Report's Detail object.

4.8.5.3.1.1 ColumnCount

Determines the number of columns of the Report's **Detail** Section. The width of each column must be equal to the Report's printable area, divided by the number of columns. The default value of this property is 1.

4.8.5.3.1.2 ColumnDirection

The **ColumnDirection** property determines the printing direction of the **Detail** Section columns. The available options are the following:

Available options for ColumnDirection

OPTION	DESCRIPTION
0 - ddCDDownAcross	Prints each **Detail** Section column from top to bottom, and then moves to the next column on the right.
1 - ddCDAcrossDown	Prints each **Detail** Section column from right to left, and so on.

The layout is determined according to the configured option:

ddCDDownAccross

ddCDAccrossDown

4.8.5.3.1.3 ColumnSpacing

◩ Determines the **Detail** Section column space. The default value of this property is 0 (zero).

4.8.5.3.1.4 KeepTogether

Determines if Report Sections are printed as a single block, on the same page. The available options for this property are the following:

Available options for KeepTogether

OPTION	DESCRIPTION
0 - ddGrpNone	There is a page break after the Report.
1 - ddGrpFirstDetail	The Report prints the **Detail** Section on the same page or column.

4.8.5.3.1.5 NewColumn

Inserts a new column break before or after printing the Section on the Report. The available options for this property are the following:

Available options for NewColumn

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in the Section.

OPTION	DESCRIPTION
1 - ddNBefore	Starts printing the Section on a new page.
2 - ddNPAfter	Starts a new page after printing the Section.
3 - ddNPBeforeAfter	Starts printing on a new page, and a new page after printing the Section.

4.8.5.3.1.6 NewPage

Inserts a page break on the Report. The available options for this property are the following:

Available options for NewPage

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in the Section.
1 - ddNBefore	Starts printing on a new page.
2 - ddNPAfter	Starts a new page after printing the Section.
3 - ddNPBeforeAfter	Starts printing on a new page, and starts a new page after printing the Section.

4.8.5.4 Group Footer

This section contains information about **properties** of the Report's Group Footer object. This object does not have events nor methods associated to it.

4.8.5.4.1 Properties

This section contains information about the properties of the Report's Group Footer object.

4.8.5.4.1.1 ColumnLayout

Determines whether the **GroupFooter** uses the same column layout configured on the **Detail** Section. If this property is set to True, the number of columns of the **Detail** Section reflects in the linked **GroupHeader** or **GroupFooter**. Otherwise, the default configuration remains. The default value of this property is True.

4.8.5.4.1.2 KeepTogether

Determines whether Report Sections are printed as a single block, on the same page. The available options for this property are the following:

Available options for KeepTogether

OPTION	DESCRIPTION
0 - ddGrpNone	There is a page break after the Report.

OPTION	DESCRIPTION
1 - ddGrpFirstDetail	The Report prints the **Detail** Section on the same page or column.

4.8.5.4.1.3 NewColumn

Inserts a new column break before or after printing the Report Section. The available options for this property are the following:

Available options for NewColumn

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break on the Section.
1 - ddNBefore	Starts printing the Section on a new page.
2 - ddNPAfter	Starts a new page after printing the Section.
3 - ddNPBeforeAfter	Starts printing on a new page, and a new page after printing the Section.

4.8.5.4.1.4 NewPage

Inserts a page break on the Report. The available options for this property are the following:

Available options for NewPage

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in the Section (default).
1 - ddNBefore	Starts printing on a new page.
2 - ddNPAfter	Starts printing a new page after printing the Section.
3 - ddNPBeforeAfter	Starts printing a new page and starts a new page after printing the Section.

4.8.5.4.1.5 PrintAtBottom

Determines whether the **Group Footer** or the Report footer are printed at the page bottom. If this property is set to True, and the Report has a page footer, the **Group Footer** and the Report footer are printed above the **Page Footer** Section. Configuring more than one Section to print the Report footer makes the following Sections to print on separate pages.

4.8.6 Objects

This section contains information about properties of Report objects. These objects do not have events nor methods associated to them.

4.8.6.1 Common Properties

This section contains information about common properties of the Report objects.

4.8.6.1.1 BackColor

Specifies a object's background color. The effect of this property is only visible if the **BackStyle** property is set to **1 - ddBKNormal**. The default value of this property is white (**RGB(255, 255, 255)**).

NOTE: This property is only valid for the Line and Page Break objects.

4.8.6.1.2 BackStyle

Specifies the object's background style. The available options for this property are the following:

- **0 - ddBKTransparent**: transparent (displays the color defined by the Section's **BackColor** property)
- **1 - ddBKNormal**: normal (displays the color defined by the object's **BackColor** property)

The default value is **0 - ddBKTransparent**.

NOTE: This property is not valid for Line, Barcode, Page Break, and Frame objects.

4.8.6.1.3 height

9 This property determines the object's height on the Report.

NOTE: This property is not valid for Line and Page Break objects.

4.8.6.1.4 left

9 Returns the object's left positioning value on the Report.

NOTE: This property is not valid for Line and Page Break objects.

4.8.6.1.5 Name

A Determines the object's name. The default value of this property is an empty **String**.

4.8.6.1.6 Tag

A Returns the Tag type linked to the object, that is, **Boolean**, **String**, **Integer**, etc.

4.8.6.1.7 Top

9 Returns the object's top value.

NOTE: This property is not valid for the Line object.

4.8.6.1.8 Visible

◩ Determines the object's visibility on the Report. If this option is set to True, the object is visible on the Report. Otherwise, the object is hidden. The default value of this property is True.

NOTE: This property is not valid for Page Break object.

4.8.6.1.9 Width

9 This property determines the object's width on the Report.

NOTE: This property is not valid for Line and Page Break objects.

4.8.6.2 Barcode

This section contains information about **properties** of the Report's Barcode object. This object does not have events nor methods associated to it.

4.8.6.2.1 Properties

This section contains information about the properties of the Report's Barcode object.

4.8.6.2.1.1 Alignment

Determines the object's text alignment on the Report. The available options are the following:

- **0 - ddtxLeft**: aligns the text to the left of the object
- **1 - ddtxRight**: aligns the text to the right of the object
- **2 - ddtxCenter**: aligns the text to the center of the object

4.8.6.2.1.2 BarWidth

9 Determines the Barcode's bar width. Configuring this width to 1 (one), the object's bar is expanded in 15 points, and so on. The higher the number configured on this property, the bigger the Barcode's bar width. The default value of this property is 1.

4.8.6.2.1.3 Caption

A Contains the object's text itself. The default value of this property is an empty **String**.

4.8.6.2.1.4 CaptionPosition

Specifies the text position of the object's **Caption** property. The available options for this property are the following:

- **0 - ddbcCaptionNone**: the text of the **Caption** property does not appear on the Barcode
- **1 - ddbcCaptionAbove**: the text of the **Caption** property appears above the Barcode
- **2 - ddbcCaptionBelow**: the text of the **Caption** property appears below the Barcode

The default value of this property is **0 - ddbcCaptionNone**.

4.8.6.2.1.5 DataField

A Configures or returns data linked to the object. This linked data may be a Database table field provided by the Query object, a mathematical expression with Query fields and VBScript functions (in this case, this field must be preceded by an equal symbol), or an E3 Tag or property. In this case, the current variable value is displayed when performing a print. The default value of this property is an empty **String**.

NOTE: The server must be running to capture the value of a variable.

4.8.6.2.1.6 Direction

Determines the Barcode's spatial orientation. The available options for this property are the following:

Available options for Direction

OPTION	DESCRIPTION
0 - ddbcLeftToRight	The Barcode orientation is from left to right.

OPTION	DESCRIPTION
1 - ddbcRightToLeft	The Barcode orientation is from right to left.
2 - ddbcTopToBottom	The Barcode orientation is from top to bottom.
3 - ddbcBottomToTop	The Barcode orientation is from bottom to top.

The default value of this property is **0 - ddbcLeftToRight**.

4.8.6.2.1.7 EnableCheckSum

Enables or disables the reading of a checksum value (character from a Barcode object). If this property is set to False, only codes with a checksum are affected.

4.8.6.2.1.8 Font

Determines the object's font in the **Caption** property. The default value of this property is "Arial".

NOTE: This property cannot be used on scripts or Links, being edited only via Studio.

4.8.6.2.1.9 Forecolor

Specifies the object's foreground color. In scripts, use the VBScript's **RGB** method to build the color to be associated to this property. The default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.2.1.10 Style

Determines the Barcode style. The available options are the following:

Available options for Style

OPTION	DESCRIPTION
0 - ddbcNone	Default Barcode style.
1 - ddbcAnsi39	ANSI 3 of 9 (Code 39). Uses letters, numbers, -,*, $, /, +, %, etc.
2 - ddbcAnsi39x	Extended ANSI 3 of 9 (Extended Code 39). Uses full ASCII characters.
3 - ddbcCode_2_of_5	2 of 5. Uses only numbers.
4 - ddbcCode25intlv	Interleaved 2 of 5. Uses only numbers.
5 - ddbcCode25mat	Matrix 25.
6 - ddbcCode39	Code, uses letters, numbers, -, *, $, /, +, %, etc.

OPTION	DESCRIPTION
7 - ddbcCode39x	Extended Code 39. Uses full ASCII characters.
8 - ddbcCode_128_a	128 A. Uses numbers, punctuation, or letters.
9 - ddbcCode_128_b	128 B. Uses strings, numbers, punctuation, or letters.
10 - ddbcCode_128_c	128 C. Uses only numbers.
11 - ddbcCode_128auto	128 Automatic. Uses full ASCII characters. Automatically selects codes between 128 A, B, and C to set the smallest Barcode value.
12 - ddbcCode_93	Code 93. Uses letters, numbers, -, *, $, /, +, %, etc.
13 - ddbcCode_93x	Extended Code 93. Uses full ASCII characters.
14 - ddbcMSI	MSI Code. Uses only numbers.
15 - ddbcPostNet	PostNet. Uses only numbers with a digital verification.
16 - ddbcCodabar	Codabar. Uses A, B, C, D, +, -, :, /, or numbers.
17 - ddbcEAN_8	EAN-8. Uses only numbers (seven numbers and digital verification).
18 - ddbcEAN_13	EAN-13. Uses only numbers (12 numbers and digital verification).
19 - ddbcUPC_A	UPC-A. Uses only numbers (11 numbers and digital verification).
20 - ddbcUPC_EO	UPC-E1. Uses only numbers. Used for UPC zero-compression symbols. In the **Caption** property, users can enter six digits of a UPC-E code or 11 digits. If an 11 digits code is inserted, the Barcode converts it to UPC-E six digits, if possible. Otherwise, converts from 11 to six UPC-E digits and nothing is displayed.
21 - ddbcUPC_E1	UPC-E1. Uses only numbers. The width of input data of UPC E1 is six number characters.
22 - ddbcRM4SCC	Royal Mail RM4SCC. Uses only letters and numbers (with digital verification). This Barcode is used in Britain.
23 - ddbcUCCEAN128	UCC/EAN_128. Uses full ASCII characters. The special version of the Code 128 is used on the HIBC application.

4.8.6.3 Ellipse, Rectangle, and Round Rectangle

This section contains information about **properties** of the Report's Ellipse, Rectange, and Round Rectangle objects. These objects do not have events nor methods

associated to them.

4.8.6.3.1 Properties

This section contains information about the properties of the Report's Ellipse, Rectangle and Round Rectangle objects.

4.8.6.3.1.1 LineColor

Specifies the object's line color. The default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.3.1.2 LineStyle

Determines the object's line style. The available options for this property are the following:

Available options for LineStyle

OPTION	DESCRIPTION
0 - ddLSTransparent	The object's line becomes transparent.
1 - ddLSSolid	The object's line appears as solid.
2 - ddLSDash	The object's line becomes dashed.
3 - ddLSDot	The object's line becomes dotted.
4 - ddLSDashDot	The object's line has dashes and dots.
5 - ddLSDashDotDot	The object's line becomes dotted, dashed, and dotted.

The default value of this property is **1 - ddLSSolid**.

4.8.6.3.1.3 LineWeight

Specifies the object's line width. Setting this width to 1, the object's line is expanded in 15 points, if set to 2, the object's line is expanded in 30 points, and so on. The higher the number set in this property, the bigger the object's width. The default value of this property is 1.

4.8.6.3.1.4 Shape

Allows changing the object's shape. The available options are the following:

Available options for Shape

OPTION	DESCRIPTION
0 - ddSHRectangle	Rectangular shape.
1 - ddSHEllipse	Elliptical or circular shape.
2 - ddSHRoundRect	Round rectangle shape.

4.8.6.4 Picture

This section contains information about **properties** of the Report's Picture object. This object does not have events nor methods associated to it.

4.8.6.4.1 Properties

This section contains information about the properties of the Report's Picture object.

4.8.6.4.1.1 DataField

A Configures or returns object's associated data. This associated data may be a Database table field provided by the Query object, a mathematical expression with Query fields and VBScript functions (in this case, this field must be preceded by the equal symbol), or an E3 Tag or property. In this case the current variable value is displayed when printing. The default value of this property is an empty **String**.

NOTE: The server must be running to capture this variable's value.

4.8.6.4.1.2 Forecolor

Specifies the object's foreground color. In scripts, use the VBScript's **RGB** function to build the color to be associated to this property. The default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.4.1.3 hyperLink

A Specifies the *link* set to the text. To use this resource, use the **OnHyperLink** event. The default value of this property is an empty **String**.

4.8.6.4.1.4 LineColor

Specifies the object's line color. The default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.4.1.5 LineStyle

Determines the object's line style. The available options for this property are the following:

Available options for LineStyle

OPTION	DESCRIPTION
0 - ddLSTransparent	The object's line becomes transparent.
1 - ddLSSolid	The object's line appears solid.

OPTION	DESCRIPTION
2 - ddLSDash	The object's line becomes dashed.
3 - ddLSDot	The object's line becomes dotted.
4 - ddLSDashDot	The object's line becomes dashed and dotted.
5 - ddLSDashDotDot	The object's line becomes dashed, dotted, and dotted.

The default value of this property is **1 - ddLSSolid**.

4.8.6.4.1.6 LineWeight

9 Specifies the object's line width. Configuring this width to 1 expands the object's line by 15 points, and so on. The higher the number configured in this property, the larger the object's width. The default value of this property is 1.

4.8.6.4.1.7 Picture

A Specifies the object's picture file. The allowed extensions are .bmp, .gif, .jpg, .cur, .ico, .emf, and .wmf. The default value of this property is an empty **String**.

4.8.6.4.1.8 PictureAlignment

Determines the alignment of object's picture. The available options are the following:

Available options for PictureAlignment

OPTION	DESCRIPTION
0 - ddPATopLeft	Aligns the object's picture on top left.
1 - ddPATopRight	Aligns the object's picture on top right.
2 - ddPACenter	Aligns the object's picture on center.
3 - ddPABottomLeft	Aligns the object's picture at the bottom left.
4 - ddPABottomRight	Aligns the object's picture at the bottom right.

The default value of this property is **2 - ddPACenter**.

4.8.6.4.1.9 SizeMode

Specifies the object's size. The available options are the following:

Available options for SizeMode

OPTIONS	DESCRIPTION
0 - ddsMClip	Displays the object in its current size.
1 - ddsMStretch	Adjusts the object according to its area.
2 - ddsMZoom	Adjusts the object image's height or width inside the specified area, without distorting it.

4.8.6.5 SetPoint

This section contains information about **properties** of the Report's SetPoint object. This object does not have events nor methods associated to it.

4.8.6.5.1 Properties

This section contains information about the properties of the Report's SetPoint object.

4.8.6.5.1.1 Alignment

Determines the object's text alignment. The available options for this property are:

- **0 - Left**: left alignment (default)
- **1 - Right**: right alignment
- **2 - Center**: centered alignment

4.8.6.5.1.2 CanGrow

Determines the application of a *stretch* effect to the object's text. In case the object's width or height are enlarged, the text accompanies this variation. If this property is set to True, the text accompanies object's height and width variations. Otherwise, this object gets its initial settings. The default value of this property is True.

4.8.6.5.1.3 CanShrink

Determines the application of a shrinking effect to the object's text. In case of object's width or height are diminished, the text accompanies this variation. If this property is set to True, text accompanies object's height and width variations. Otherwise, this object gets its initial settings. The default value of this property is

True.

4.8.6.5.1.4 ClassName

A Returns the object's class. This is a read-only property.

4.8.6.5.1.5 DataField

A Configures or returns object's associated data. This associated data can be a Database table field provided by the Query object, a mathematical expression with Query fields and VBScript functions (in this case, this field must be preceded by an equal symbol), or an E3 Tag or property. In this case the current variable value is displayed when printing. The default value of this property is an empty **String**.

NOTE: The server must be running to capture variable's value.

4.8.6.5.1.6 Font

A This property determines object's text font. The default value of this property is an empty **String**. This property cannot be used in scripts or Links, and it is configured only via Studio.

4.8.6.5.1.7 ForeColor

The **ForeColor** specifies the object's background color. In scripts, use the VBScript's **RGB** function to build the color to be associated to this property. The default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.5.1.8 hyperLink

A The **hyperLink** property determines the *link* to be set to the text. To use this resource, use the **OnHyperLink** event.

4.8.6.5.1.9 Multiline

The **Multiline** property indicates whether the text has multiple lines (True) or is a simple Text Box (False). This can be viewed when the Viewer object is running. The default value of this property is False.

4.8.6.5.1.10 OutputFormat

A Configures or returns text format of the **Text** property, used on one of the format functions (**FormatCurrency**, **FormatDateTime**, **FormatNumber**, and **FormatPercent**) in Visual Basic.

4.8.6.5.1.11 Style

A Returns the object's text style. This is a read-only property.

4.8.6.5.1.12 SummaryDistinctField

A Determines the field name to be used by the selected function in the **SummaryFunc** property. This property is only valid if the function defined in the **SummaryFunc** property is from the **Distinct Summary** function group, which encompasses functions from 9 to 15, and when the **SummaryType** property value is different from 0.

4.8.6.5.1.13 SummaryFunc

Determines the function type to be used for processing field values specified in the **DataField** property, as described on the next table. This property is only valid when the **SummaryType** property value is different from 0.

Available options for SummaryFunc

OPTION	DESCRIPTION
0 - Sum	Calculates the sum of all values inside the specified total intervals (group, page, or Report).
1 - Avg	Calculates the average of all values inside the specified total intervals (group, page, or Report).
2 - Count	Counts the number of all values inside the specified total intervals (group, page, or Report).
3 - Min	Displays the lowest value (minimum value) of all values inside the specified total intervals (group, page, or Report).
4 - Max	Displays the highest values (maximum value) of all values inside the specified total intervals (group, page, or Report).
5 - Var	Calculates the variance of the values inside the specified total intervals (group, page or Report).
6 - VarP	Calculates the population variance of the values inside the specified total intervals (group, page or Report).
7 - Dev	Calculates the deviation of the values inside the specified total intervals (group, page or Report).
8 - DevP	Calculates the population deviance of the values inside the specified total intervals (group, page or Report).

OPTION	DESCRIPTION
9 - DSum	Calculates the sum of all distinct values inside the specified total intervals (group, page or Report).
10 - DAvg	Calculates the average based on distinct values inside the specified total intervals (group, page or Report).
11 - DCount	Counts the number of distinct values inside the specified total intervals (group, page or Report).
12 - DVar	Calculates the variance of the distinct values inside the specified total intervals (group, page or Report).
13 - DVarP	Calculates the population variance of the distinct values inside the specified total intervals (group, page or Report).
14 - DDev	Calculates the deviance of the distinct values inside the specified total intervals (group, page or Report).
15 - DDevP	Calculates the population deviance of the distinct values inside the specified total intervals (group, page or Report).

4.8.6.5.1.14 SummaryGroup

A This property is only valid when the **SummaryType** property is equal to **3 - SubTotal**. The **SummaryGroup** property indicates what is the **GroupHeader** Section name used to control subtotals, that is, at each change in the **GroupHeader** value, sum is then restarted.

NOTE: When this property is used, the **CanShrink** and **CanGrow** properties are disabled.

4.8.6.5.1.15 SummaryRunning

Determines if a summary running total is performed, according to the following options:

- **0 - None**: does not perform the summary
- **1 - Group**: calculates the summary running for each total interval specified
- **2 - All**: calculates the summary running of all values in the Report, despite of grouping

This property is only valid when the **SummaryType** property is different from 0.

4.8.6.5.1.16 SummaryType

Determines the summary type or level to be generated. The available options are the following:

Available options for SummaryType

OPTION	DESCRIPTION
0 - None	No summary generation.
1 - GrandTotal	Specifies that the whole Report's content is summarized.
2 - PageTotal	Specifies that a subtotal by page is generated.
3 - SubTotal	Specifies that a subtotal for each group is generated, defined in the **SummaryGroup** property.
4 - PageCount	Specifies a page counter.

4.8.6.5.1.17 Text

A Determines the text to be set to the object.

4.8.6.5.1.18 VerticalAlignment

Determines the object's vertical alignment, as follows:

- **0 - Top**: top alignment
- **1 - Middle**: middle alignment
- **2 - Bottom**: bottom alignment

4.8.6.5.1.19 WordWrap

Enables or disables a line break in the text, in case the available text area overrides the limits determined by the object. For this property to work, the **Multiline** property must be equal to True. If it is True, the `white-space:nowrap` configuration is displayed in the **Style** property.

4.8.6.6 Text

This section contains information about **properties** of the Report's Text object. This object does not have events nor methods associated to it.

4.8.6.6.1 Properties

This section contains information about the Report's Text object.

4.8.6.6.1.1 Alignment

The **Alignment** property determines the text alignment, as follows:

- **0 - ddtxLeft**: left alignment
- **1 - ddtxRight**: right alignment
- **2 - ddtxCenter**: centered alignment (default value)

4.8.6.6.1.2 Angle

The **Angle** property indicates the text's angle. This property's value must be specified in decimal degrees, that is, for the text to be displayed in a 45 degrees angle, its value must be equal to 450. The default value of this property is 0 (horizontal positioning).

4.8.6.6.1.3 Caption

The **Caption** property contains the object's text. The default value of this property is an empty **String**.

4.8.6.6.1.4 ClassName

The **ClassName** property allows specifying a global CSS class (indicated by an external *Cascade Style Sheet*) to be applied to the text. A CSS class is a format pattern that determines the letter type and size, paragraph alignment and spacing, among other features. Using CSS, users can apply a pre-defined format pattern to a text, unifying and speeding up text presentation. To apply a specific style, users can use the **Style** property. The default value of this property is the Normal style.

4.8.6.6.1.5 Font

This property indicates the text's font name. The default value of this property is an empty **String** (E3 uses the system's default font).

NOTE: This property cannot be used in scripts or Links, it is only configured via Studio, and can only be changed at run time.

4.8.6.6.1.6 ForeColor

The **Forecolor** property specifies the object's background color. In scripts, use the VBScript's **RGB** function to build the color to be associated to this property. The default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.6.1.7 hyperLink

A The **hyperLink** property determines a *link* to be set to the object. To use this resource, use the **OnHyperLink** event. The default value of this property is an empty **String**.

4.8.6.6.1.8 Multiline

◩ The **Multiline** property indicates whether the text has multiple lines (True) or is a simple Text Box (False). This can be viewed when the Viewer object is running. The default value of this property is False.

4.8.6.6.1.9 Style

A The **Style** property allows specifying a CSS (*Cascade Style Sheet*) style for the text, replacing the global style. This property's value must be a valid CSS **String**, otherwise this property is ignored. The default value of this property is an empty **String** (E3 uses the system's default style). Example:

```
Sub Report1_OnBeforePrint
  Label1.Style = "font-family: Times; font-weight: bold;_
    text-align: center; color: RGB(255, 255, 0)"
End Sub
```

4.8.6.6.1.10 VerticalAlignment

The **VerticalAlignment** property determines the text's vertical alignment, as follows.

- **0 - ddTxTop**: top alignment (default)
- **1 - ddTxMiddle**: centered alignment
- **2 - ddTxBottom**: bottom alignment

4.8.6.6.1.11 WordWrap

◩ Enables or disables a line break on the text, in case the available text area overrides the object's limits. For this property to work, the **Multiline** property must be True.

4.8.6.7 Line

This section contains information about **properties** of the Report's Line object. This object does not hae events nor methods associated to it.

4.8.6.7.1 Properties

This section contains information about the properties of the Report's Line object.

4.8.6.7.1.1 LineColor

The **LineColor** property specifies the object's line color. The default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.7.1.2 LineStyle

This property determines the object's line style. The default value of this property is **1 - ddLSSolid**. The other available options for this property are the following:

Available options for LineStyle

OPTION	DESCRIPTION
0 - ddLSTransparent	The object's line becomes transparent.
1 - ddLSSolid	The object's line appears solid.
2 - ddLSDash	The object's line becomes dashed.
3 - ddLSDot	The object's line becomes dotted.
4 - ddLSDashDot	The object's line becomes dashed and dotted.
5 - ddLSDashDotDot	The object's line becomes dashed, dotted, and dotted.

4.8.6.7.1.3 LineWeight

The **LineWeight** property specifies the object's line width. Configuring this width to 1, the object's line expands by 15 points; if this width is 2, the object's line expands by 30 points, and so on. The higher the number configured in this property, the larger the object's width. The default value of this property is 1.

4.8.6.7.1.4 X1

The **X1** property enables or disables the position of the line's starting point in the X axis.

4.8.6.7.1.5 X2

The **X2** property determines the position of the line's ending point in the X axis. The default value of this property is empty.

4.8.6.7.1.6 Y1

The **Y1** property determines the position of the line's starting point in the Y axis. The default value of this property is empty.

4.8.6.7.1.7 Y2

9 The **Y2** property determines the position of the line's ending point in the Y axis. The default value of this property is empty.

4.8.6.8 Page Break

This section contains information about the **properties** of the Report's Page Break object. This object does not have events nor methods associated to it.

4.8.6.8.1 Properties

This section contains information about the properties of the Report's Page Break object. This object does not have events nor methods associated to it.

4.8.6.8.1.1 Enabled

◩ The **Enabled** property enables or disables the object on the Report. If this option is set to True, the object is enabled on the Report. Otherwise, the object is disabled. The default value of this property is True.

4.8.6.9 Frame

This section contais information about **properties** of the Report's Frame object. This object does not have events nor methods associated to it.

4.8.6.9.1 Properties

This section contains information about the properties of the Report's Frame object.

4.8.6.9.1.1 CanGrow

◩ Determines the application of a *stretch* effect to the object's text. In case the object's width or height are enlarged, the text accompanies this variation. If this property is set to True, the text accompanies object's height and width variations. Otherwise, this object gets its initial settings. The default value of this property is True.

4.8.6.9.1.2 CanShrink

◩ Determines the application of a shrinking effect to the object's text. In case the object's width or height are diminished, the text accompanies this variation. If this property is set to True, the text accompanies the object's height and width variations. Otherwise, this object gets its initial settings. The default value of this property is True.

4.8.6.9.1.3 CloseBorder

This property enables or disables viewing the border line of the Frame's base, in case it overrides more than one page on the Report.

4.8.6.9.1.4 left

The **left** property returns the object's left position on the Report. The default value of this property is empty.

4.8.6.10 E3Chart

This section contains information about the **properties** of the Report's E3Chart object.

4.8.6.10.1 Properties

Usage example of E3Chart properties on a Report

The following script must be created on the Report's **Page Header** Section, using the **OnBeforePrint** event. Example:

```
Sub OnBeforePrint
  Set chart = _
    Report.Sections("PageHeader").Controls("E3Chart1")
  chart.LoadData()
  chart.FitAll
End Sub

Sub OnBeforePrint
  ' This script copies the configuration from the
  ' E3Chart object to the E3Chart chart
  ' which is used in the report.
  Set chartfrom = _
    Application.GetFrame().Screen.Item("E3Chart1")
  Set chart = _
    Report.Sections("PageHeader").Controls("E3Chart2")
  chart.CopyConfig(chartfrom)
  chart.LoadData()
  chart.FitAll
End Sub
```

NOTE: Descriptions for E3Chart properties on a Report are the same explained in the topic **Properties**.

CHAPTER 5

Server Objects

This section contains information about common properties of Server objects. These objects do not have common events nor methods. Objects sharing Server properties are the following:

- **Server's Runtime Objects**
- **Configuration-Only Objects**
- **Drivers**
- **Data Server**
- **Database**
- **Historic**
- **Storage**
- **Formulas**
- **Alarms**

5.1 Common Properties

This section contains information about common properties of all Server objects.

5.1.1 IsAlarmArea

◩ Enables or disables the Alarm Area functionality for Server objects. This property is not available for the following objects: Alarm Area, Alarm Source, and Alarm Server. The default value of this property is False. When enabling this property, the object then has the same properties of an **Alarm Area**.

5.1.2 Common Properties of Server Objects as Alarm Areas

This section contains information about common properties of Server objects that behave as Alarm Areas.

5.1.2.1 ActiveAlarms

9 Determines the number of active alarms inside an object. This is a read-only property and its default value is 0 (zero).

5.1.2.2 ActiveHighAlarms

Indicates the number of active alarms with high severity. This is a read-only property.

5.1.2.3 ActiveHighNACKAlarms

Indicates the number of non-acknowledged alarms with high severity. This is a read-only property.

5.1.2.4 ActiveLowAlarms

Indicates the number of active alarms with low severity. This is a read-only property.

5.1.2.5 ActiveLowNACKAlarms

Indicates the number of non-acknowledged alarms with low severity. This is a read-only property.

5.1.2.6 ActiveMedAlarms

Indicates the number of active alarms with medium severity. This is a read-only property.

5.1.2.7 ActiveMedNACKAlarms

Indicates the number of non-acknowledged alarms with medium severity. This is a read-only property.

5.1.2.8 ActiveNACKAlarms

Indicates the number of non-acknowledged alarms inside an object. This is a read-only property.

5.1.2.9 Alarm

Indicates that there are active alarms inside the object. If this option is set to True, there is at least one active alarm inside the object, and the property **ActiveAlarms** performs a reading from server, then indicating the amount of active alarms. Otherwise, property **ActiveNACKAlarms** performs a reading of non-acknowledged alarms. This is a read-only property.

5.1.2.10 AlarmVerify

◩ Enables a check on all alarms inside the object. After enabling that check (True), if the value of property **ActiveAlarms** is greater than 0 (zero), server then checks active alarms, as well as non-acknowledged ones, listing these last ones in the property **ActiveNACKAlarms**. This property is useful to avoid a cascade effect on some systems, where the occurrence of an event triggers a large amount of related alarms.

5.1.2.11 InactiveHighNACKAlarms

9 Indicates the number of active and unacknowledged alarms with high severity. This is a read-only property.

5.1.2.12 InactiveLowNACKAlarms

9 Indicates the number of active and unacknowledged alarms with low severity. This is a read-only property.

5.1.2.13 InactiveMedNACKAlarms

9 Indicates the number of active and unacknowledged alarms with medium severity. This is a read-only property.

5.1.2.14 InactiveNACKAlarms

9 Determines the total amount of active and unacknowledged alarms. This is a read-only property.

5.1.2.15 UserFields

🔗 Returns an object that is a collection of Alarm's User Fields of a Server object. Please check the item **Collection of Alarm's User Fields** for more information about the collection of objects returned by this property.

5.2 Collection of Alarm's User Fields

This section contains information about **methods** and **properties** common to the collection of Alarm's User Fields of E3's Server objects, returned by the **UserFields** property of **Alarm Areas** and **Alarm Sources**.

5.2.1 Common Methods

This section contains information about methods common to the collection of Alarm's User Fields of E3's Server objects.

5.2.1.1 Item

Item(*Index*)

Returns an Alarm's User Field object indicated by the Index parameter, which can be the value of the **Index** property (integer) or the value of the **Name** property (text) of the object.

5.2.2 Common Properties

This section contains information about the properties common to the collection of Alarm's User Fields of E3's Server objects.

5.2.2.1 Count

9 Returns the number of child objects (items) of a collection of Alarm's User Fields. If the Collection has no child objects, then this property returns 0 (zero).

5.2.3 Alarm's User Fields

This section contains information about **properties** of objects of type Alarm's User Field in the Collection of Alarm's User Fields returned by the UserFields property of **Alarm Areas**, **Alarm Sources**, and **Server Objects** that behave as Alarm Areas. This object does not have events nor methods associated to it.

5.2.3.1 Properties

This section contains information about the properties of objects of type Alarm's User Field in the Collection of Alarm's User Fields.

5.2.3.1.1 Index

9 Returns the index of this object in the Collection of Alarm's User Fields. This value can be used as a parameter for the Collection's **Item** method.

5.2.3.1.2 Link

A Returns the Link configured for this Alarm's User Field. To change at run time the Link configured in this property, users must deactivate the object. For example:

```
Dim sAlarm
Set sAlarm = _
  Application.GetObject("ConfigAlarms.Area.DigitalAlarm1")
sAlarm.Deactivate()
sAlarm.UserFields.Item("BatchName").Link = _
  "Driver.TagBatchName.Value"
sAlarm.Activate()
```

5.2.3.1.3 Name

A Returns the name of this object in the Collection of Alarm's User Fields. This value can be used as a parameter for the Collection's **Item** method.

5.2.3.1.4 Value

v Returns or configures the current value of the Alarm's User Field. This property has a different behavior depending on the object being active or inactive. With the object active, the value returned when reading this property obeys the following priorities:

1. If there is a forced value (the **ValueSource** property equal to **evsForcedValue**), returns this value
2. If there is a configured Link (the **ValueSource** property equal to **evsLink**), returns the Link's current value
3. Searches for the value of this User Field in the hierarchically superior Area (traverses the hierarchy of Areas in ascending order)
4. If there is no hierarchically superior Area with a forced value or a configured Link for this User Field, retrieves the default value of the User Field configured in the Alarm Server

If the object is inactive, reading this property returns a forced value, in case it exists (the **ValueSource** property equal to **evsForcedValue**). If there is no forced value, reading this property fails.

The behavior of writings to this property is the same, whether it is active or inactive. The writing fails if there is a configured Link (the **ValueSource** property equal to **evsLink**). Otherwise, the new value is accepted and the **ValueSource** property is automatically configured to **evsForcedValue**.

5.2.3.1.5 ValueSource

Specifies the source of the **Value** property of this Alarm's User Field. Possible values for this property are the following:

- **0 - evsInherited**: The **Value** property is inherited from the Alarm Server or from the parent Area (default value)
- **1 - evsLink**: The **Value** property is provided by the **Link** property
- **2 - evsForcedValue**: The **Value** property is provided by the user

This property accepts writings with the object active, as well as inactive. In both cases, it is not possible to write the value **evsLink**. To change the **ValueSource** property to **evsLink**, users must write directly to the **Link** property (deactivating the

object if at run time).

NOTE: When exporting an Alarm's User Field object, the **ValueSource** property must be placed after the **Link** and **Value** properties in the order of columns.

5.3 Server's Runtime Objects

This section contains information about object available only at run time, the Server (Application) and the Application Folders.

5.3.1 Server

This section contains information about specific **methods** of the Server (Application) object. This object has the **Item** and **Save** general methods, besides the **Name** and **Count** properties, which are described in the **General Events, Methods, and Properties of Objects**. This object does not have events associated to it.

5.3.1.1 Methods

This section contains information about specific methods of the Server object.

NOTE: Methods described here can only be used at run time, they are not available in Studio.

5.3.1.1.1 ClearFailure

ClearFailure(*FailureName*)

This method is called to indicate that a failure reported by **ReportFailure** becomes inactive. The *FailureName* parameter specifies a failure name (user-defined) and must be passed to the **ReportFailure** method when that method is called.

5.3.1.1.2 E3GetActor

E3GetActor()

This method returns the logon name of the user who started the current request in E3Run. If E3Run is not currently handling an operation generated by another process, returns "System". If there is no user logged on the process which generated the current request, returns "Anonymous".

5.3.1.1.3 LoadReport

LoadReport(*ReportName*)

Loads a Report model. See Viewer's **LoadReport** method for an example of this

method's usage.

NOTE: The Report's **PrintPreview** method is not available in Reports loaded using this method.

5.3.1.1.4 ReportFailure

ReportFailure(*FailureName*, *FailureDescription*, *FailureWeight*)

Allows the application to report failures to the Server. The reported failures are usually conditions that keep the application from working partially, such as errors or malfunctions, and normally cannot be detected by the Server. Thus, this method can be used in these situations:

- To advise the system operator about problems in the server
- To help Hot-Standby manager to elect the best server to take control of the application

NOTE: The **ReportFailure** method is only available in the Server.

The **ReportFailure** method has the following parameters:

ReportFailure method parameters

NAME	DESCRIPTION
FailureName	Failure name (user-defined). This parameter must be passed to **ClearFailure** method whenever the failure becomes inactive. For example, "FaultCOM1".
FailureDescription	Failure description (user-defined). For example, "Communication failure at COM1".
FailureWeight	Defines failure's severity (or importance). A zero value indicates no gravity. Higher values indicate more severe faults.

Example:

```
Sub TagSerialStatus_OnValueChanged()
  If Value Then
    ' The tag value is True, application is on error
    Application.ReportFailure "FAULT_COM1", _
      "Communication fault at COM1", 100
  Else
    ' The tag value is False, clears the failure
    Application.ClearFailure "FAULT_COM1"
  End If
```

```
End Sub
```

5.3.1.1.5 Trace

Trace(*MessageText*[, *LogTimeStamp*[, *BreakLine*]])

Allows recording messages in a text file. Messages will be recorded in a file with the same name and path as the Domain file, but as a .txt file. Each new message is always inserted at the end of the file. In case there is some failure in file recording (like access denied, for example), there will be a script error.

This method can be used, for example, to record script debugging messages not executed at Viewer (because in this case it is not possible to use **MsgBox** method).

Trace method parameters

NAME	DESCRIPTION
MessageText	User-defined text message.
LogTimeStamp	(Optional) a Boolean indicating whether each record must have date and time (timestamp). If omitted, assumes True.
BreakLine	(Optional) a Boolean indicating whether there is a line break at the end of each message. If False, all records in the file are in a single line. If omitted, assumes True.

5.3.1.1.6 TrackEvent

TrackEvent(*EventMessage*, *Comment*, *TimeStamp*)

This method allows manually generating events via script. See Viewer's **TrackEvent** event for more information.

5.3.2 Application Folder

The Application Folder object is similar to **Data Folder** object, from **Data Server**, allowing to group Server objects inside folders. However, there are important differences:

- The Application Folder object is only available at run time
- This object has **Application**, **Count**, **Name**, **Parent**, and **PathName** general properties, and also **Item** and **Save** general methods
- This object allows using **For Each** VBScript command to enumerate its child objects, but that kind of access only takes into account Server objects, not listing, for example, screens and resources inside folders
- The root folder from where it is possible to enumerate objects is always the **Server** object

5.4 Configuration-Only Objects

This section contains information about objects which can be only used at configuration time.

5.4.1 E3StudioApplication

This section contains information about specific **methods** of the E3StudioApplication object. This object does not have specific events nor properties.

5.4.1.1 Methods

This section contains information about specific methods of the E3StudioApplication object.

NOTE: Methods described here can be only used in Studio, they are not available at run time.

5.4.1.1.1 CreateFile

CreateFile(*ProjectName*, *ClassName*[, *FileName*, *FolderName*, *RunWizard*, *OpenView*])

Creates a new object on a project file (.prj). Parameters for this method are the following:

- **ProjectName**: the name of the project file where the object will be inserted. This parameter cannot be empty, and the project file must be loaded on Studio. It can be project file's full path, or a Domain file's relative path. There is no need to add the .prj extension
- **ClassName**: The name of the object's class to be created. See the table below for possible values for this parameter
- **FileName**: The name of the inserted object. If this parameter is omitted, then it will be used a default name to create an object of the type indicated by *ClassName*. The maximum allowed size for this parameter is 32 characters, and if it already exists, it is automatically incremented
- **FolderName**: The name of the Folder where the object will be created. If this parameter is omitted, the object is then created at project file's root. The Folder name must obey the same rules of **Note**'s section of the **RenameFolder** method
- **RunWizard**: A **Boolean** indicating whether the configuration wizard of the class indicated by *ClassName* will be opened after executing this method. The default value of this parameter is True
- **OpenView**: A **Boolean** indicating whether the editor of the inserted object will be

opened after executing this method. The default value of this parameter is True

Possible values for ClassName parameter

OBJECT TO CREATE	PARAMETER VALUE
Alarm Configuration	DB.AlarmConfig
Alarm Server	DB.AlarmServer
Database	DB.DBServer
Data Server	Panel.DataServer
Formula	DB.Formula
FrameSet	Panel.FrameSet
Historic	DB.Hist
I/O Driver	IODrv.IODriver
OPC Driver	IODrv.OPCDriver
Report	Panel.Report
Screen	Panel.Screen
Storage	DB.Historian
Viewer	Panel.Viewer

5.4.1.1.2 CreateFolder

CreateFolder(*ProjectName*, *FolderName*[, *ParentFolder*])

Creates a new Folder with the name defined by *FolderName*, inside project file (.prj) *ProjectName*, which parent object is indicated by *ParentFolder*. The following restrictions apply to this method:

- The project file indicated in parameter *ProjectName* must exist, and it must be opened on Studio
- If parameter *FolderName* is empty, then a default name will be used to create the Folder. For example, `Folder1`
- The maximum allowed size for parameter *FolderName* is 32 characters
- If the name indicated in parameter *FolderName* already exists, then it will be automatically incremented
- The name of the Folder in *FolderName* must obey the same rules of **Note**'s section of the **RenameFolder** method
- The Folder indicated in parameter *ParentFolder* must exist. If this parameter is omitted, the new Folder is then created at project file's root

5.4.1.1.3 CreatePRJ

CreatePRJ(*Filename*)

Creates a new project file (.prj) named *Filename*, and then adds it to the current

Domain, if it exists. If the parameter *Filename* is not an absolute path, then it will use a path relative to the Domain, or the default path for project files (saved on Windows Registry). The .prj extension is added automatically, if needed.

5.4.1.1.4 RenameFolder

RenameFolder(*OldName*, *NewName*)

Allows renaming a Folder in an open project file in Studio, whether it belongs to the Domain or not. The *OldName* parameter is the full path to the Folder to rename, and the *NewName* parameter is the new name, without a path.

NOTES:

- If Folder's full path in the parameter *OldName*, or the new name in the parameter *NewName* starts with a character that is not a letter (with no diacritical marks or "ç"), or if it has any character that is not a letter (with no diacritical marks or "ç"), a number (0-9), or an underscore, you must surround that name with brackets.
- If the parameter *NewName* is invalid, or if the parameter *OldName* has a syntax error, or else the path indicated by *OldName* is not found on any open project file, a script error will occur
- Password-protected project files must be unlocked before using this method

5.5 Drivers

This section contains information about events, methods and properties of the following objects: **I/O Driver**, **I/O Tag**, **I/O Block**, **I/O Block Element**, **OPC Driver**, **OPC Tag**, **OPC Block**, **OPC Block Element**, and **OPC UA Driver**.

5.5.1 I/O Driver

This section contains information about **events**, **methods** and **properties** of the I/O Driver object.

5.5.1.1 Events

This section contains information about the events of the I/O Driver object.

5.5.1.1.1 AfterStart

AfterStart()

Occurs after the I/O Driver has started to communicate. It is common to make a script for this event using the **Write** method to perform equipment setup. Example:

```
Sub Driver1_AfterStart()
  ' After started, communication writes values
  ' to the equipment or device
  Write 0, 2, 55, 2, 33.4
  Write 0, 3, 55, 20, "Metal"
```

```
End Sub
```

5.5.1.1.2 AfterStop

AfterStop()

Occurs after the I/O Driver has finished the communication. Use the **AfterStop** event to perform any necessary actions after Driver communication has finished.

5.5.1.1.3 BeforeStart

BeforeStart()

Occurs when the Driver is about to start the communication. Use the **BeforeStart** event to perform any necessary actions before starting communication, such as setting up Driver parameters. Example:

```
Sub Driver1_BeforeStart()
  ' It performs the startup of driver parameters before
  ' starting the communication
  P1 = 0
  P2 = 20
  P3 = 80
  P4 = 0
End Sub
```

5.5.1.1.4 BeforeStop

BeforeStop()

Occurs when the Driver is about to finish the communication. Use the **BeforeStop** event to perform any necessary actions before finishing communication, such as writing to or reading values from the equipment or device, before communication is no longer available.

5.5.1.1.5 OnCommError

OnCommError(*EvtType*, *Size*, *Element*, *N1*, *N2*, *N3*, *N4*)

Occurs when an I/O Driver detects any writing or reading error. Use an **OnCommError** event to check when a writing or reading failure occurred in the I/O Driver. Event variables receive information about this error. With these values, it is possible to track back Tags with I/O problems.

OnCommError event variables

NAME	DESCRIPTION
EvtType	Informs which type of operation the Driver was performing when the error occurred, according to the following options: • **0**: Single Element reading error (**Size** = 1). **Param1** is N1, **Param2** is N2, **Param3** is N3, and **Param4** is N4 • **1**: Single Element writing error (**Size** = 1). **Param1** is N1, **Param2** is N2, **Param3** is N3, and **Param4** is N4 • **2**: Block reading error (I/O Block). **Size** is determined by the number of Block Elements. **Param1** is N1, **Param2** is N2, **Param3** is N3, and **Param4** is N4. • **3**: Block writing error (I/O Block). **Size** is determined by the number of Block Elements. **Param1** is N1, **Param2** is N2, **Param3** is N3, and **Param4** is N4.
Size	Number of values being read or written.
Element	Index of the Element being read or written inside a Block.
N1	First parameter of a reading or writing operation that generated the error.
N2	Second parameter of a reading or writing operation that generated the error.
N3	Third parameter of a reading or writing operation that generated the error.
N4	Fourth parameter of a reading or writing operation that generated the error.

Example:

```
Sub Driver1_OnCommError(Type, Size, Element, N1, N2, N3, N4)
  Application.GetObject("Data.InternalTag1").Value = _
    Application.GetObject("Data.InternalTag1").Value + 1
  Application.GetObject("Data.EvtType").Value = EvtType
  Application.GetObject("Data.Size").Value = Size
  Application.GetObject("Data.Element").Value = Element
  Application.GetObject("Data.N1").Value = N1
  Application.GetObject("Data.N2").Value = N2
  Application.GetObject("Data.N3").Value = N3
  Application.GetObject("Data.N4").Value = N4
End Sub
```

5.5.1.1.6 OnCommErrorEx

OnCommErrorEx(*ErrorInfo*)

Occurs soon after the execution of the **OnCommError** event.

ErrorInfo parameter information

NAME	DESCRIPTION
ErrorInfo.EvtType	Indicates the type of operation that caused the error: • **0**: Tag reading • **1**: Tag writing • **2**: Block reading • **3**: Block writing
ErrorInfo.Size	Size of the Block that caused the error (if it is a Tag, **Size** is equal to 1).
ErrorInfo.Element	Index of the Block Element that caused the error.
ErrorInfo.Nx	N*x* or B*x* parameters (*x* = 1, 2, 3, 4) of the operation that caused the error.
ErrorInfo.ParamDevice	**ParamDevice** parameter (**String**) of the operation that caused the error.
ErrorInfo.ParamItem	**ParamItem** parameter (**String**) of the operation that caused the error.

5.5.1.1.7 OnTagRead

OnTagRead(*Tag*)

Occurs on Tag reading, whenever the I/O Driver returns a new value or an error. This means that if Tag value or Tag quality do not change, the event will not be triggered. For this event to work, the **EnableDriverEvent** property must necessarily be enabled. In addition, the **PercentDeadband** property can also influence the event, if the **EnableDeadband** property is enabled. Example:

```
Sub Tags_OnTagRead(Tag)
  Set Obj = Application.GetObject("Data1.TagName")
  Obj.Value = Tag.Name
  Set Obj = Application.GetObject("Data1.TagRead")
  Obj.Value = True
  Set Obj = Application.GetObject("Data1.TagType")
  Obj.Value = TypeName(Tag)
End Sub
```

5.5.1.1.8 OnTagWrite

OnTagWrite(*Tag*, *Succeeded*, *User*)

Occurs when a writing operation is triggered on any Driver Tag.

OnTagWrite event variables

NAME	DESCRIPTION
Tag	A reference to the Tag object being written. For example, users can access the Tag property using a syntax such as **Tag.DocString**.
Succeeded	A **Boolean** value that indicates whether a writing operation was successful or not.
User	Parameter that receives the name of the user performing the writing operation. The default value of this parameter is "System". If there is no user logged in the Viewer generating this event, this parameter contains the value "Anonymous".

5.5.1.2 Methods

This section contains information about the methods of the I/O Driver object.

5.5.1.2.1 Write

Write(*N1*, *N2*, *N3*, *N4*, *Value*[, *WriteSyncMode*])

Writes data to the device. This method returns a **Boolean** indicating whether the operation was successful or not. The *N1* to *N4* parameters correspond to Driver's **N** parameters. The *Value* parameter defines the value to be written to the Driver. The *WriteSyncMode* parameter allows users to use a writing mode other than the one used in the Driver. The available options for this parameter are:

- **0**: Uses the writing mode configured in the Driver (default)
- **1**: Executes a synchronous writing
- **2**: Executes an asynchronous writing (no confirmation)

If the WriteSyncMode parameter is omitted, the writing mode configured in the Driver is also used. For more information about these parameters, please check the Driver's documentation.

5.5.1.2.2 WriteEx

WriteEx(*N1*, *N2*, *N3*, *N4*, *Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes data to the equipment. This method returns a **Boolean** indicating whether the operation has been successful or not. *N1* to *N4* parameters correspond to Driver's *[N]* parameters. The *Value* parameter defines the value to be written to the Driver. For further information on these parameters, see the Driver's documentation.

The *Timestamp*, *Quality*, and *WriteStatus* parameters are optional. When omitted, this method's behavior is similar to the **Write** method. The *Timestamp* parameter specifies the time stamp to be written to the Tag (if supported by the equipment). If omitted, the time stamp from the moment of the writing operation is assumed. The *Quality* parameter indicates quality (from 0 to 255). If omitted, a Good (192) quality is assumed. The *WriteStatus* parameter receives a value returned by the Driver, indicating the writing status (if supported by the Driver, according to its own documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are:

- **0**: write mode configured in the Driver
- **1**: synchronous write mode
- **2**: asynchronous write mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver will also be used. Example:

```
Dim status
If tag001.WriteEx(0, 0, 0, 0, 100, , , status) Then
  MsgBox "Successful writing, status = " & status
Else
  MsgBox "Writing failed, status = " & status
End If
```

5.5.1.3 Properties

This section contains information about the properties of the I/O Driver object.

5.5.1.3.1 DisableIOServerPool

◩ Forces the Driver to not participate in the Pool of IOServer processes, that is, the Driver runs in an exclusive IOServer. The default value of this property is False.

NOTE: For more information, please check the topic **Pool of IOServer Processes**, in the **User's Manual**.

5.5.1.3.2 DriverLocation

A Defines what Driver will be used by the I/O Driver object to communicate with a device. This property accepts a **String** with the complete Driver's path, in case it is not in the same Domain's directory, or the relative Driver's path if it is in the same Domain's directory. In case there is no open Domain on Studio, a relative path is considered as starting from the folder where the project or library containing the Driver object is. After that, the **DriverName** property will change to the Driver's

description. This property cannot be modified once communication has started. Default value is an empty **String**.

NOTE: It is always advisable to use the option **Browse DLL**, on I/O Driver object's contextual menu, to set this property correctly.

5.5.1.3.3 DriverName

A Contains a **String** describing the Driver associated to the I/O Driver object. For this, users must first set the **DriverLocation** property. This is a read-only property, and it is not available at run time.

5.5.1.3.4 EnableReadGrouping

◩ Enables reading optimization (automatic Tag grouping). This property cannot be changed at run time. Default value is True. The reading optimization only happens if the Driver supports this feature.

5.5.1.3.5 P1

9 Use the **P1** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
End Sub
```

5.5.1.3.6 P2

9 Use the **P2** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
End Sub
```

5.5.1.3.7 P3

9 Use the **P3** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
End Sub
```

5.5.1.3.8 P4

9 Use the **P4** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
  P4 = 500
End Sub
```

5.5.1.3.9 ParamDevice

A Defines the address of the device accessed by the Driver. This property is inherited by the Driver's Blocks and Tags, which can override this value, if necessary.

5.5.1.3.10 ReadRetries

9 Indicates the number of Driver's reading retries, in case of error. If it is set to 2, for example, it means the Driver will retry failed communications twice, apart from the original try.

5.5.1.3.11 ShareMaximum

9 Defines the maximum number of I/O Drivers that will be grouped into a shared I/O Server. This property is only used when the **ShareServer** property is enabled. Example:

```
' This driver will not be shared
ShareServer = False
ShareMaximum = <any value>
```

```
' All drivers are grouped into the same IOServer
' Does not define a limit
ShareServer = True
ShareMaximum = 0
' Groups every 5 drivers into an IOServer
ShareServer = True
ShareMaximum = 5
```

5.5.1.3.12 ShareServer

◩ If True, this Driver will share its execution among all other I/O Drivers that have the same **String** in the **DriverLocation** property. It means that only the first I/O Driver that is set will execute the communication initialization procedure. All other shared I/O Drivers will ignore all setting parameters from *P1* to *P4*, as well as other settings. Otherwise, if the **ShareServer** property is False, the Driver will not share any kind of communication with other I/O Drivers. This property cannot be modified once communication has started. Default value is False.

5.5.1.3.13 WriteFeedbackMode

◩ Allows controlling writing feedback confirmation to Tags. It applies only to readable Tags, that is, Tags whose **AllowRead** property is set to True. Through this property, reading Tags that receive writings is quickier. This property has the following setting options:

Available options for WriteFeedbackMode

OPTION	DESCRIPTION
0 - wfWaitNextRead	Tag reading is performed normally on the next scan.
1 - wfImmediateReadAfterWrite	After each writing, a confirmation reading will be performed as soon as possible.
2 - wfTrustWriteSuccess	If the Driver indicates a successful writing, the written value is directly assumed by the Tag, without reading it from the PLC.

The default value is **1 - wfImmediateReadAfterWrite**. Applications created on earlier versions, before this property existed, assume 0 when loaded. Example:

```
Sub CommandButton1_Click()
  Dim mode
  mode = Application.GetObject("Driver1").WriteFeedbackMode
  MsgBox mode
  Select case mode
  Case 0
    MsgBox "Tag reading will be done in the next scan."
  Case 1
    MsgBox "After each writing, a confirmation _
      reading will be done as soon as possible."
```

```
  Case 2
    MsgBox "If the driver indicates writing success, _
      the written value is assumed directly by the tag, _
      without reading it from the PLC."
  End Select
End Sub
```

NOTE: When a value of 2 is used, time stamp and quality may be incorrect, since in a successful writing the value is assumed by the Tag without searching for time stamp and quality in the PLC. In addition, the assumed value itself may present a little deviation due to any kind of simplification in numbers that may occur in the Driver or in the PLC. Also, note that some Drivers or protocols may indicate success, even when the writing has failed. So, the other values (**0 - wfWaitNextRead** or **1 - wfImmediateReadAfterWrite**) should be preferred when possible.

5.5.1.3.14 WriteRetries

9 Indicates the number of Driver's writing retries, in case of error. If it is set to 2, for example, it means the Driver will retry a failed communication twice, apart from the original try.

5.5.1.3.15 WriteSyncMode

Determines how writings will be sent to the I/O Server (synchronous or asynchronous mode). This property has the following configuration options:

Available options for WriteSyncMode

OPTION	DESCRIPTION
0 - wsmDefault	Synchronous mode (default).
1 - wsmSync	Synchronous mode. Every time a value is written on a Tag, E3Run sends the writing to the I/O Server and waits for the return of the writing.
2 - wsmAsyncUnconfirmed	Asynchronous mode without confirmation. All writings are sent to the I/O Server without waiting its return, and it is always assumed that the writing was successful. When in asynchronous mode, the Tag's writing methods (**Write**, **WriteEx**) always return True immediately, and the writing status (on methods returning this status) remains always empty. The Driver's **OnTagWrite** event is executed as soon as the writing is sent to the I/O Server, and the *Succeeded* parameter always remain in True.

Asynchronous writings will be executed by the I/O Server as soon as the Driver is available (when current reading is over). If several asynchronous writings are sent to the I/O Server, the Driver will only return the readings after all asynchronous writings are executed.

5.5.1.4 I/O Block

This section contains information about **events**, **methods**, and **properties** of the I/O Block object.

5.5.1.4.1 Events

This section contains information about the events of the I/O Block object.

5.5.1.4.1.1 OnRead

OnRead()

Occurs when the Driver performs an I/O Block reading. Use the **OnRead** event when it is necessary to perform some operation soon after some data has been modified in the I/O Block object, such as the **Quality**, **TimeStamp** or even the **Value** property of some of the Block's Element. Example:

```
Sub IOBlock1_OnRead()
  ' When reading a block, assign to the InternalTag1 tag
  ' the value of the block element elm1.
  Set obj=Application.GetObject("DataServer1.InternalTag1")
  Set elm = Application.GetObject("Driver1.IOBlock1.elm1")
  obj.Value = elm.Value
End Sub
```

5.5.1.4.2 Methods

This section contains information about the methods of the I/O Block object.

5.5.1.4.2.1 Write

Write([*WriteSyncMode*])

Writes the I/O Block's current value in the equipment. Usually, this script command is used only when the object's **AllowWrite** property is False.

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are:

- **0**: writing mode configured in the Driver
- **1**: synchronous writing mode

- **2**: asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver will also be used. For further information, see the Driver's documentation. This method returns a **Boolean** indicating whether the operation has been successful or not.

5.5.1.4.2.2 WriteEx

WriteEx(*Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes values to the equipment. All parameters are optional; if omitted, method's behavior is the same as the **Write** method. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *Value* parameter defines the value to be written to the Driver. Data type depends on the Driver; if omitted, Tag's current value is assumed. The *Timestamp* parameter specifies the time stamp to be written to the Tag (if supported by the equipment). If omitted, the time stamp from the moment of the writing operation is assumed. The *Quality* parameter indicates quality (from 0 to 255). If omitted, a Good (192) quality is assumed. The *WriteStatus* parameter receives a value returned by the Driver, indicating writing status (if supported by the Driver, according to its documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are:

- **0**: writing mode configured in the Driver
- **1**: synchronous writing mode
- **2**: asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver will also be used. Example:

```
Sub Tag1_OnRead()
  ' Use WriteEx to transfer values from one driver to another
  Application.GetObject("Driver2.Tag")._
    WriteEx Value, TimeStamp, Quality
End Sub
```

5.5.1.4.3 Properties

This section contains information about the properties of the I/O Block object.

5.5.1.4.3.1 AdviseType

Controls **Advise** mode. The available options are:

Available options for AdviseType

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The Tag is updated if the **AllowRead** property is True.
1 - AdviseWhenLinked	The Tag is updated only if the **AllowRead** property is True, and the Tag is associated to an active object (that is, one Display in an open Screen, an enabled Alarm, etc.). A Tag link for this purpose can be assigned to these properties: **Value**, **RawValue**, **Quality** and from **Bit00 to Bit31** for Block Elements, and **Quality** and **TimeStamp** for I/O Blocks.

5.5.1.4.3.2 AllowRead

Defines whether the Block will be read by the I/O Driver. If True, the Driver automatically updates the I/O Elements inserted in the Block, in time spans defined by the **Scan** property. Otherwise, the I/O Block will neither be read nor updated. This property can be modified at run time. Default value is True. Example:

```
Sub Button1_Click()
  'Stops block reading
  Set obj = Application.GetObject("Driver1.block1")
  obj.AllowRead = False
End Sub
```

5.5.1.4.3.3 AllowWrite

Defines whether the Block will be written automatically when the **Value** property from its I/O Block Elements is modified. If True, the modifications will be sent to the device associated to the I/O Driver. Otherwise, the modifications will be ignored. I/O Block Elements will not accept values if this property is set to False, unless the **AllowRead** property is also set to False. Example:

```
Sub Button1_Click()
  'Disables block writing
  Set obj = Application.GetObject("Driver1.block1")
  obj.AllowWrite = False
End Sub
```

5.5.1.4.3.4 B1

Specifies the data set in the device this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once

communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
End Sub
```

5.5.1.4.3.5 B2

Specifies the data set in the device this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
End Sub
```

5.5.1.4.3.6 B3

Specifies the data set in the device this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
End Sub
```

5.5.1.4.3.7 B4

Specifies the data set in the device this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
  B4 = 524
End Sub
```

5.5.1.4.3.8 EnableDeadBand

Enables or disables the **PercentDeadBand** property of Block Elements. If True, the Block value is only updated when it changes, and its new value exceeds the limit defined by the **PercentDeadBand** property of any Block Element. Otherwise, the Block is always updated, and the dead band limit is not checked. Whenever possible, users should keep the dead band enabled, because it enhances data acquisition and

the processing performance. Usually, dead band is disabled only for Tags that return values representing events that need to be handled on the **OnRead** event. The default value of this property is True.

NOTES:

- In case there are more than one Block Element mapped to the same index, the dead band configuration used is the one that results in the smallest absolute dead band value
- If any Block index has an unmapped Element, the dead band in this index is equal to 0, that is, any variation in the value of the Element violates the dead band
- If the **EnableDeadBand** property is enabled, the last value sent to the Block is compared to the current value read, Element by Element. If any Block Element violates its dead band, the whole Block is updated

5.5.1.4.3.9 EnableDriverEvent

Controls the way **OnTagRead** events are generated, which occur in the I/O Driver that contains the Block. If True, the **OnTagRead** event generation is then enabled by this Tag. Otherwise, it is disabled. The three kinds of I/O Elements (I/O Tags, I/O Blocks, and Block Elements) can generate this event. The event occurs in the Driver, not in the Block.

5.5.1.4.3.10 ParamDevice

Defines the address of the device accessed by the Block. This property is inherited from the Driver, but its value can be overridden, if necessary.

5.5.1.4.3.11 ParamItem

Identifies data accessed by the Block from the device.

5.5.1.4.3.12 Quality

Informs the quality of the value contained in the **Value** property. Each time the Driver sets a new value to the Block, it also sets data quality. This is a read-only property. Default value is 0 (Bad Quality).

NOTE: For further information about quality, see the topic **Quality** on **E3 User's Manual**.

5.5.1.4.3.13 Scan

Specifies the duration of the scan used by the server to update the Block. This property is represented in milliseconds, and can be modified after communication has started. It is used only when the **AllowRead** property is set to True. When users set this property in several Blocks in the application, they should increase the value

of this property for those Blocks that do not vary very much in the device, enabling other Blocks with higher priority to be read more frequently, thus enhancing general application performance. Default value is 1,000 (1 second). Scan value must be greater than zero. Example:

```
Sub IOBlock1_BeforeStart()
  Scan = 152
End Sub
```

5.5.1.4.3.14 Size

Defines the size of value sets in this Block. See the Driver documentation for this property's limits, according to *B1* to *B4* parameters. By creating Elements for the Block, users can access reading and writing values for the device. This property cannot be modified once communication has been started. Default value is 0. Example:

```
Sub IOBlock1_BeforeStart()
  Size = 10
End Sub
```

5.5.1.4.3.15 TimeStamp

Updated whenever there is a change in value or status in either the **Value** or the **Quality** properties. Informs time stamp associated to both value and quality in an I/O Block. This is a read-only property. Default value is 00:00:00.

5.5.1.4.4 I/O Block Element

This section contains information about **methods** and **properties** of the I/O Block Element object. This object does not have events associated to it.

5.5.1.4.4.1 Methods

This section contains information about the methods of the I/O Block Element object.

Write

Write([*WriteSyncMode*])

Writes I/O Block Element's current value to the equipment. Usually, this script is used only when the object's **AllowWrite** property is False.

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are:

- **0**: write mode configured in the Driver
- **1**: synchronous writing mode

- **2**: asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver will also be used. For further information, see the Driver's documentation. This method returns a **Boolean** indicating whether the operation has been successful or not.

WriteEx

WriteEx(*Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes values to the equipment. All its parameters are optional; if omitted, method's behavior is the same as the **Write** method. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *Value* parameter defines the value to be written to the Driver. Data type depends on the Driver; if omitted, Tag's current value is assumed. The *Timestamp* specifies the time stamp to be written to the Tag (if supported by the equipment). If omitted, the time stamp from the moment of the writing operation is assumed. The *Quality* indicates quality (from 0 to 255). If omitted, a Good (192) quality is assumed. The *WriteStatus* receives a value returned by the Driver, indicating writing status (if supported by the Driver, according to its own documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are:

- **0**: writing mode configured in the Driver
- **1**: synchronous writing mode
- **2**: asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver will also be used. Example:

```
Sub Tag1_OnRead()
  ' Use WriteEx to transfer values from one driver to another
  Application.GetObject("Driver2.Tag")._
    WriteEx Value, TimeStamp, Quality
End Sub
```

5.5.1.4.4.2 Properties

This section contains information about the properties of the I/O Block Element object.

Bit00 to Bit31

These properties represent the 32 bits in I/O Tag's **Value** property, ranging from **Bit00** (the least significant bit) to **Bit31** (the most significant bit). By modifying each one of these bits, users will modify Tag's **Value** property, and vice versa, but this

only happens when the **UseBitFields** property is True. Default value is False.

NOTE: Bit values (from **Bit00** to **Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

DeviceHigh

9 Defines the highest value reached by the Block Element in the device. This property is used in the calculation of the **PercentDeadBand** property, and also to adjust the device's scale value before setting it to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is set to True. Default value is 1000.

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by the scale adjustment, that is, they represent bits from the value read by the equipment before conversion.

DeviceLow

9 Defines the lowest value reached by the Block Element in the device. This property is used in the calculation of the **PercentDeadBand** property, and also to adjust the device's scale value before setting it to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is set to True. Default value is 0.

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by the scale adjustment, that is, they represent bits from the value read by the equipment before conversion.

EnableDriverEvent

◩ Controls how the **OnTagRead** event is generated, which occurs in the I/O Driver that contains the Block. If this Tag's **EnableDriverEvent** property is configured to True, generating the **OnTagRead** event is enabled by this Tag. Otherwise, it is disabled. The three kinds of I/O Elements (I/O Tag, I/O Block, and Block Element) can generate this event. The event occurs in the Driver, and not in the Block.

EnableScaling

◩ If True, all values from the device will receive scale adjustments according to the **DeviceHigh**, **DeviceLow**, **EUHigh** and **EULow** properties before they are attributed to the **Value** property. Otherwise, no adjustment in scale will be made. Default value is False. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
```

```
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EU

A Identifies the engineering unit represented by the value, such as degrees, meters, etc. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EUHigh

9 Defines the highest value to be attributed to the **Value** property, adjusting the scale to the value from the device before it is done. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is True. The default value of this property is 1,000. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

EULow

9 Defines the lowest value to be attributed to the **Value** property, adjusting the scale

to the value from the device before it is performed. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is True. The default value of this property is 0. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

Index

9 Specifies this Element's position among the other Elements configured by the I/O Block's **Size** property where it is inserted. This property's value range from 0 to the value defined by **Size** minus 1. (For example: If **Size** is 20, **Index** ranges from 0 to 19). This property can be modified once communication has started. Default value is 0, but when mapping the Elements from a Block, Studio automatically adjusts **Index** parameter to the specified value. Example:

```
Sub Element_OnStartRunning()
  Index = 15
End Sub
```

PercentDeadBand

9 The **PercentDeadBand** property determines the minimum variation (dead band) of a Block Element, so that this value can be updated in E3. This value is specified as a percentage of the difference between the **DeviceHigh** and the **DeviceLow** properties. This property is only used if the **EnableDeadBand** property of the Block is set to True. If the **PercentDeadBand** property is equal to 0, the Block Element does not have a dead band, and any variation in its value is passed to E3. Otherwise, E3 receives a new value only if its difference, relative to the current value, is larger than the dead band. The default value of this property is 0.

Quality

9 Informs the quality of the value contained in the **Value** property. Each time the Driver attributes a new value to the Tag, it also sets data quality. This property is read-only. Default value is 0 (Bad Quality).

NOTE: For further information on quality, see the topic **Quality** on **E3 User's Manual**.

RawValue

It is the Element's original value, before **EnableScaling** property has acted upon it. So, if **EnableScaling** is False, the **Value** and **RawValue** properties will behave identically.

UseBitFields

If True, every time the value from **Value** property changes, the bits referring to **Bit00 to Bit31** properties will be updated. Also, every time the value from **Bit00 to Bit31** properties are modified, the value from **Value** property is updated and sent to the device, if the **AllowWrite** property is True. Otherwise, bits will not undergo any modification. This property can be updated once communication has started. Default value is False.

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

Value

Updated whenever a new valid reading from a device is performed, using *N1* to *N4* parameters. Data type (**Integer**, **Floating Point**, **String**) depends on the Driver to which the Element is associated and its parameters.

This property is updated this way if the **AllowRead** property is True, and when there are no communication errors (in this case, the **Quality** and **TimeStamp** properties are updated), according to a scan time defined in the **Scan** property. Another way to use this property is to write values in the device; for this, just set a new value to the **Value** property or to some of the bits from **Bit00 to Bit31**. In this case, the **AllowWrite** property must be True.

This is also I/O Tag's default property. So, it is not mandatory for a value reference to an I/O Tag to make the **Value** property explicit, in order to access its value. Default value is empty (no value). Example:

```
Sub Button1_Click()
  ' Accesses a tag and shows current value
  ' tag1 is an I/O tag
  Set obj = Application.GetObject("IODriver1.tag1")
  MsgBox "Tag1's current value: " & obj.Value
  'This can also be done in a different way,
  'with no need to show Value property, which is default
  MsgBox " Tag1's current value: " & obj
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

5.5.1.5 I/O Folder

This section contains information about **properties** of the I/O Folder object of the I/O Driver. This object does not have events nor methods associated to it.

5.5.1.5.1 Properties

This section contains information about the properties of the I/O Driver's I/O Folder object.

5.5.1.5.1.1 ParamDevice

A Defines the address of the device accessed by the I/O Folder. This property is inherited from the I/O Driver, and its value can be overwritten, if needed.

5.5.1.6 I/O Tag

This section contains information about **events**, **methods** and **properties** of the I/O Tag object.

5.5.1.6.1 Events

This section contains information about the events of the I/O Tag object.

5.5.1.6.1.1 OnRead

OnRead()

Occurs when a Tag reading is performed by the Driver. Use the **OnRead** event when it is necessary to perform some operation soon after some data has been modified in the Tag, such as the **Value**, **Quality** or **TimeStamp** properties. This event is generated by a background reading. Example:

```
Sub CommTag1_OnRead()
  ' When reading the tag, assign its value
  ' to the InternalTag1 tag.
  Set obj = Application.GetObject("DataServer1.InternalTag1")
  obj = Value ' CommTag1 Value
End Sub
```

5.5.1.6.2 Methods

This section contains information about the methods of the I/O Tag object.

5.5.1.6.2.1 Write

Write([*WriteSyncMode*])

Writes I/O Tag's current value to the equipment. Usually, this method is used only when the object's **AllowWrite** property is False.

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are:

- **0**: write mode configured in the Driver
- **1**: synchronous writing mode
- **2**: asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver will also be used. For further information, see the Driver's documentation. This method returns a **Boolean** indicating whether the operation has been successful or not.

5.5.1.6.2.2 WriteEx

WriteEx(*Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes values to the equipment. All parameters are optional; if omitted, method's behavior is the same as the **Write** method. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *Value* parameter defines the value to be written to the Driver. Data type depends on the Driver; if omitted, Tag's current value is assumed. The *Timestamp* specifies the time stamp to be written to the Tag (if supported by the equipment). If omitted, the time stamp from the moment of the writing operation is assumed. The *Quality* parameter indicates quality (from 0 to 255). If omitted, a Good (192) quality is assumed. The *WriteStatus* parameter receives a value returned by the Driver, indicating writing status (if supported by the Driver, according to its documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are:

- **0**: writing mode configured in the Driver
- **1**: synchronous writing mode
- **2**: asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver will also be used. Example:

```
Sub Tag1_OnRead()
  ' Use WriteEx to transfer values from one driver to another
  Application.GetObject("Driver2.Tag")._
    WriteEx Value, TimeStamp, Quality
End Sub
```

5.5.1.6.3 Properties

This section contains information about the properties of the I/O Tag object.

5.5.1.6.3.1 AdviseType

Controls the **Advise** mode. The available options are:

Available options for AdviseType

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The Tag is kept updated if the **AllowRead** property is equal to True.
1 - AdviseWhenLinked	The Tag is only updated if the **AllowRead** is equal to True and it is linked to any active object, for example, a Display on an open Screen, an enabled alarm, among others. The Tag link for this purpose can be performed on the following properties: **Value**, **RawValue**, **TimeStamp**, **Quality**, and from I/O Tag's **Bit00 to Bit31**.

Example:

```
Sub CommandButton3_Click()
  MsgBox Application._
    Application.GetObject("Driver1.Tag1").AdviseType
End Sub
```

5.5.1.6.3.2 AllowRead

Defines whether the Tag will be read by the I/O Driver. If True, the Driver automatically updates **Value** and **Bits** (from **Bit00** to **Bit31**) properties, in time spans defined by the **Scan** property. Otherwise, the I/O Tag will neither be read nor updated. This property can be modified at run time. Default value is True. Example:

```
Sub Button1_Click()
  ' Stops tag reading
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowRead = False
End Sub
```

5.5.1.6.3.3 AllowWrite

◩ Defines whether the Tag will be written automatically when **Value** or **Bits** (from **Bit00** to **Bit31**) properties are modified. If True, the modifications will be sent to the device associated to the I/O Driver. Otherwise, the modifications will be ignored. Default value is True. Example:

```
Sub Button1_Click()
  ' Disables tag writing
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowWrite = False
End Sub
```

5.5.1.6.3.4 Bit00 to Bit31

◩ These properties represent the 32 bits in I/O Tag's **Value** property, ranging from **Bit00** (the least significant bit) to **Bit31** (the most significant bit). By modifying each one of these bits modifies Tag's **Value** property, and vice versa, but this only happens when the **UseBitFields** property is True. Default value is False.

NOTE: Bit values (from **Bit00** to **Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

5.5.1.6.3.5 DeviceHigh

9 Defines the highest value achieved by this Tag in the device. Adjusts the device's value scale before being attributed to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is True. Default value is 1,000. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

5.5.1.6.3.6 DeviceLow

9 Defines the lowest value achieved by this Tag in the device. Adjusts the device's value scale before being attributed to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is True. Default value is 0. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

5.5.1.6.3.7 EnableDeadBand

◩ Enables or disables the **PercentDeadBand** property. If True, Tag value is only updated when it changes, and its new value exceeds the limit defined by the **PercentDeadBand** property. Otherwise, the Tag is always updated, and the dead band limit is not checked. Whenever it is possible, users should keep dead band enabled, because it enhances data acquisition and processing performance. Usually, users should disable dead band only for Tags that return values representing digital or analogical events, and it is necessary to process these events via script on Tag's **OnRead** event. Default value is True.

5.5.1.6.3.8 EnableDriverEvent

◩ Controls how the **OnTagRead** event is generated, which occurs in the I/O Driver that contains the Block. If this Tag's **EnableDriverEvent** property is configured to True, then this Tag's **OnTagRead** event is enabled. Otherwise, it is disabled. The three kinds of I/O Elements (I/O Tag, I/O Block, and Block Element) can generate this event. The event occurs in the Driver, and not in the Block.

5.5.1.6.3.9 EnableScaling

◩ If True, all values from the device will receive scale adjustments, according to **DeviceHigh**, **DeviceLow**, **EUHigh** and **EULow** properties before they are attributed to

the **Value** property. Otherwise, no adjustment in the scale will be performed. Default value is False. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

5.5.1.6.3.10 EU

A Identifies the engineering unit represented by the value, such as degrees, meters, etc. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

5.5.1.6.3.11 EUHigh

9 Defines the highest value to be attributed to **Value** property, adjusting the scale to the value from the device before it is performed. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is True. The default value of this property is 1,000. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

5.5.1.6.3.12 EULow

Defines the lowest value to be attributed to the **Value** property, adjusting the scale to the value from the device before it is performed. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when **EnableScaling** property is True. The default value of this property is 0. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale. That is, they represent the bits from the value read by the equipment before the conversion.

5.5.1.6.3.13 N1

Specifies the device's variable that this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once communication has started. Default value is 0. Example:

```
Sub Tag_OnStartRunning()
  N1 = 10
End Sub
```

5.5.1.6.3.14 N2

Specifies the device's variable that this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once communication has started. Default value is 0. Example:

```
Sub Tag_OnStartRunning()
  N2 = 3
End Sub
```

5.5.1.6.3.15 N3

9 Specifies the device's variable that this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once communication has started. Default value is 0. Example:

```
Sub Tag_OnStartRunning()
  N1 = 10
  N3 = 5
  N4 = 20
End Sub
```

5.5.1.6.3.16 N4

9 Specifies the device's variable that this Tag is associated to. See the Driver documentation for appropriate parameters. This property can be modified once communication has started. Default value is 0. Example:

```
Sub Tag_OnStartRunning()
  N1 = 10
  N4 = 20
End Sub
```

5.5.1.6.3.17 ParamDevice

A Defines the address of the device accessed by the Tag. This property is inherited from the Driver, but its value can be overridden, if necessary.

5.5.1.6.3.18 ParamItem

A Identifies data this Tag accesses from inside the device.

5.5.1.6.3.19 PercentDeadBand

9 Determines dead band for a Tag, so that this value can be updated on E3. This value is the variation percentage between the **DeviceHigh** and **DeviceLow** property values. This property is only used if the Tag's **EnableDeadBand** property is set to True. If **PercentDeadBand** value is 0, the Tag will not have dead band, and any variation in value will be passed to E3. Otherwise, E3 will receive a new value only if its difference is larger than dead band. Default value is 0.

5.5.1.6.3.20 Quality

9 Informs the quality of the value contained in the **Value** property. Each time the Driver attributes a new value to the Tag, it also sets data quality. This is a read-only property. Default value is 0 (Bad Quality).

NOTE: For further information on quality, see the topic **Quality** on **E3 User's Manual**.

5.5.1.6.3.21 RawValue

It is the Element's original value, before the **EnableScaling** property has acted upon it. So, if this property is False, the **Value** and **RawValue** properties will behave identically.

5.5.1.6.3.22 Scan

Specifies the duration of the scan used by the server to update the **Value** property. This property is represented in milliseconds, and can be modified after communication has started. It is used only when the **AllowRead** property is set to True. When you set this property in several application Tags, users should increase the value of the property to those Tags that do not vary very much in the device, enabling other Tags with higher priority to be read more frequently, and enhancing general application performance. Default value is 1,000 (1 second). The scan value should be above zero. Example:

```
Sub Tag_OnStartRunning()
  Scan = 1500
End Sub
```

5.5.1.6.3.23 TimeStamp

Updated whenever there is a change in value or status in either **Value** or **Quality** properties. Informs time stamp associated to both value and quality in I/O Tag. This is a read-only property. Default value is 00:00:00.

5.5.1.6.3.24 UseBitFields

If True, every time the value from **Value** property is modified, the bits referring from **Bit00 to Bit31** properties will be updated. Also, every time the value from **Bit00 to Bit31** properties are modified, the value from **Value** property is updated and sent to the device, if the **AllowWrite** property is True. Otherwise, bits will not undergo any modification. This property can be updated once communication has started. Default value is False.

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by adjusting the scale. That is, they represent the bits from the value read by the equipment before conversion.

5.5.1.6.3.25 Value

Updated whenever a new valid reading of a device is performed, using *N1* to *N4* parameters. Data type (**Integer**, **Floating Point**, **String**) depends on the Driver to which the Element is associated and its parameters.

This property is updated this way if the **AllowRead** property is True, and when there

are no communication errors (in this case, the **Quality** and **TimeStamp** properties are updated), according to a scan time defined in the **Scan** property. Another way to use this property is to write values in the device. For this, just set a new value to the **Value** property or to some of the bits from the **Bit00 to Bit31** properties. In this case, the **AllowWrite** property must be True.

This is also I/O Tag's default property. So, it is not mandatory for a value reference to a I/O Tag to make the **Value** property explicit in order to access its value. Default value is empty (no value). Example:

```
Sub Button1_Click()
  ' Accesses a tag and shows current value
  ' tag1 is an I/O tag
  Set obj = Application.GetObject("IODriver1.tag1")
  MsgBox "Tag1's current value: " & obj.Value
  'This can also be done in a different way,
  'with no need to show Value property, which is default
  MsgBox " Tag1's current value: " & obj
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by adjusting in scale. That is, they represent the bits from the value read by the equipment before conversion.

5.5.2 OPC Driver

This section contains information about **events** and **properties** of the OPC Driver object. This object does not contain methods associated to it.

5.5.2.1 Events

This section contains information about the events of the OPC Driver object.

5.5.2.1.1 OnTagRead

OnTagRead(*Tag*)

Occurs when reading a Tag, always when a new value or error is returned by the OPC Server. That is, if Tag value or quality do not change, this event is not triggered. For this event to occur, the Tag's **EnableDriverEvent** property must be enabled.

5.5.2.1.2 OnTagWrite

OnTagWrite(*Tag*, *Succeeded*, *User*)

Occurs when a writing is triggered on any OPC Driver's Tag. For this event to occur, the Tag's **EnableDriverEvent** property must be enabled. If the writing is asynchronous, the **OnTagWrite** event is triggered only when the server sends an answer indicating whether the writing was successful or not.

Parameters of the OnTagWrite event

NAME	DESCRIPTION
Tag	A reference to the Tag object being written. For example, users may access a Tag property using a **Tag.DocString** syntax.
Succeeded	A **Boolean** value indicating success or failure on writing.
User	Parameter that receives the user performing the writing operation. The default value of this parameter is "System". If there is no user logged in Viewer generating this event, this parameter contains the value "Anonymous". If the writing is asynchronous, or if there is a failure reported in an asynchronous way, this parameter always contains the value "System".

5.5.2.2 Properties

This section contains information about the properties of the OPC Driver object.

5.5.2.2.1 CallTimeout

Specifies a timeout for an answer on any call or access to an OPC server, such as writings, creation of OPC Groups, Tag creation, removal, or browsing, changes to Tag's **Advise** mode, etc. If this timeout is exceeded, E3 considers the server as locked (unavailable) and starts the reconnection process. The value of this property cannot be negative. Configuring this property with 0 (zero) disables this timeout, causing all accesses to the OPC server to take an undefined amount of time, which may lock the whole application if this access is synchronous. The default value of this property is 10000 (10 seconds). Applications created in previous versions, when loaded in the current version, assume the value 0 (zero) for compatibility reasons. This value must be carefully configured, so that it does not lock the whole application nor forces an unnecessary disconnection, if the OPC server is actually taking too long to respond to certain requests.

5.5.2.2.2 Compatibility

Controls the usage of OPC's default interfaces by E3's OPC client. The available options are:

- **0 - AnyVersion**: Normal operation (recommended). The OPC Driver communicates with both DA 2.0x and 1.0a servers (preference is given to 2.0x interfaces)
- **1 - Version10A**: Forces communication with DA 1.0a standard for servers that

support both DA 2.0x and 1.0a

- **2 - Version20**: Forces communication only with OPC DA 2.0x standard

This property cannot be modified when OPC client communication is enabled (both in Studio and at run time).

NOTE: Only as a last resource the Driver should be configured with a value different from **0 - AnyVersion** (default). This property is for advanced usage only, and applies strictly to overcome a possible incompatibility situation with some specific OPC servers.

5.5.2.2.3 ConnectionTimeout

9 Specifies a maximum timeout to establish a connection with an OPC server, including all connection steps, even steps before accessing the server itself, such as access to the **OPCENUM** service or to Windows Registry. The value of this property cannot be negative. Configuring this property with 0 (zero) disables this timeout, causing this limit to be the actual error's return time by the services needed to access the OPC server. The default value of this property is 10000 (10 seconds). Applications created in previous versions, when loaded in the current version, assume the value 0 (zero) for compatibility reasons.

5.5.2.2.4 ReconnectPeriod

9 Controls the period of connection to the OPC Server. If connection is lost, the OPC Driver stops and restarts until this action returns success. The period is configured in milliseconds, and when its value is set to 0 (zero), reconnection is disabled. Because the OPC Driver is stopped and started, **BeforeStart** and **AfterStop** events are generated. When connection is lost, all related Tags are disconnected from their current status (bad, quality, or null value). Example:

```
Sub OPCDriver1_AfterStart()
  Application.GetObject("OPCDriver1.OPCGroup1")._
    ReconnectPeriod = 0
End Sub
```

5.5.2.2.5 ServerId

A Determines the server to which the OPC Driver must connect. Although the default value of this property is empty, the OPC Driver is not going to connect. This property can only be modified when the OPC Driver is not connected. Example:

```
Sub CommandButton1_Click()
  Set Opc = Application.GetObject("OPCDriver1")
  Opc.Deactivate
  Opc.ServerId = "ElipseSCADA.OPCSvr.1"
  Opc.ServerMachine = "\\server2"
  Opc.Activate
```

```
End Sub
```

5.5.2.2.6 ServerMachine

A This property determines the address of the station where the OPC server is running. For applications running locally, this property may remain blank (default). Otherwise, users must specify the path (for example, "\\ServerName"). This property can only be modified when the OPC Driver is disconnected. Example:

```
Sub CommandButton1_Click()
  Set Opc = Application.GetObject("OPCDriver1")
  Opc.Deactivate
  Opc.ServerId = "ElipseSCADA.OPCSvr.1"
  Opc.ServerMachine = "\\server2"
  Opc.Activate
End Sub
```

5.5.2.2.7 ServerName

A This property returns the OPC server's name or description. This property is different from the **ServerID** property, which is a code. Example:

```
Sub OPCDriver1_AfterStart()
  MsgBox _
    Application.GetObject("OPCDriver1.OPCGroup1").ServerName
End Sub
```

5.5.2.2.8 ServerStatus

Determines the OPC Server's connection status. The available options are described on the next table.

Available options for ServerStatus

OPTION	DESCRIPTION
-1 - ServerStatus_Unknown	The OPC Driver is connected to the OPC server but either status is not informed or the OPC client has its **ReconnectPeriod** property set to 0 (zero).
0 - ServerStatus_NotConnected	The OPC Driver is not connected to the OPC Server. This happens when, for example, the OPC Driver is not active, or a connection has not been established yet for any reason.

The following values are only informed when the **ReconnectPeriod** property is different from 0 (zero). This time period specifying a status is retrieved from the server. In case this status is not correctly informed, this property can maintain its value in -1, or the disconnection can be detected in this case, which brings the **ServerSatus** property to 0 (zero). Values are based on the five default status types

defined for OPC servers.

Available options for ReconnectPeriod different from zero

OPTION	DESCRIPTION
1 - ServerStatus_Running	The server is running normally.
2 - ServerStatus_Failed	The server is not running. An unspecified error occurred on the server.
3 - ServerStatus_NoConfig	The server is running, but without information on its configuration.
4 - ServerStatus_Suspended	The server was temporarily suspended.
5 - ServerStatus_Test	The server is in test mode.

Example:

```
Sub CommandButton1_Click()
  Dim status
  status =  Application.GetObject("OPCDriver1").ServerStatus
  MsgBox "Driver status is " & status
  Select Case status
    Case -1
      MsgBox "OPC Driver is connected to the OPC Server_
        but its status was not informed."
    Case 0
      MsgBox "OPC Driver is not connected to an OPC Server."
    Case 1
      MsgBox "The OPC Server is running normally."
    Case 2
      MsgBox "The OPC Server is not running."
    Case 3
      MsgBox "The OPC Server is running without information_
        on its configuration."
    Case 4
      MsgBox "The OPC Server has been temporarily suspended."
    Case 5
      MsgBox "The OPC Server is running in Test Mode."
  End Select
End Sub
```

NOTE: If users want this property to behave as if it were a **Boolean**, use **ServerStatus** different from 0 (zero). This basically differentiates only between existing a connection or not, without considering more specific server statuses. In addition, this expression does not depend on using the **ReconnectPeriod** different from 0 (zero).

5.5.2.2.9 WriteFeedbackMode

This property allows controlling feedback from Tag writings. The configuration options of this property are described on the next table.

Available options for the WriteFeedbackMode property

OPTION	DESCRIPTION
0 - wfWaitNextRead	After writing, waits normally for the next reading.
1 - wfImmediateReadAfterWrite	Forces an asynchronous reading from a device right after each writing.
2 - wfTrustWriteSuccess	The written value is assumed by the Tag immediately, if the writing succeeded.

The default value of this property is **0 - wfWaitNextRead** for applications created before the implementation of this property, and **1 - wfImmediateReadAfterWrite** for applications created after its implementation.

NOTES:

- OPC Driver's **WriteFeedbackMode** property cannot be changed while this object is active.
- In the **2 - wfTrustWriteSuccess** option, for asynchronous writings, the value is assumed in the Tag right after scheduling that writing, if the operation succeeded. However, if that writing fails later, the value in the Tag may be incorrect. For synchronous writings, this value is assumed right after that writing finishes, if succeeded.
- Please check I/O Driver's **WriteFeedbackMode** property, which has a similar behavior.

5.5.2.3 OPC Group

This section contains information about **methods** and **properties** of the OPC Group object. This object does not contain methods associated to it.

5.5.2.3.1 Methods

This section contains information about the methods of the OPC Group object.

5.5.2.3.1.1 Refresh

Refresh(*Source*)

Forces the server to resend values from all OPC Group Tags that have their readings enabled, whether they have changed their values or not. The *Source* parameter determines the argument of OPC Driver's data source. If the informed value is **1** (**RefreshFromCache**), sent values are server's cache values. Otherwise, if the informed value is **2** (**RefreshFromDevice**), values are updated in server's cache before sending. For this method to work, the OPC Group's **Enable** property, as well as reading at least one Tag of the OPC Group, must be enabled. For more information about this mechanism of enabling readings (the **Advise** mode), please check Tag's **AllowRead** and **AdviseType** properties.

5.5.2.3.2 Properties

This section contains information about the properties of the OPC Group object.

5.5.2.3.2.1 BlockMode

◩ This property determines the behavior when activating or deactivating an OPC Group. when this property is configured to True, communication of Tags of the OPC Group starts at once. This usually improves performance (short activation time), by minimizing the number of calls to the OPC server. When this property is configured to False, its behavior is activating communication for each Tag of the OPC Group individually (according to the normal activation sequence of objects). With it, for example, the first Tag of the OPC Group (in the order seen in the Organizer) communicates before the last Tag. Altough slower, this activation mode of an OPC Block can be an advantage when trying to perform an operation (a Tag writing, for example) in a script of a Tag's **OnStartRunning** event. The deactivation occurs in an analogous way. When the value of this property is True, the deactivation of communication of OPC Group Tags occurs at once, when finishing the deactivation of the whole OPC Group. In case this property is configured to False, deactivation of communication occurs individually for each Tag (according to the normal activation sequence of objects).

5.5.2.3.2.2 DeadBand

9 This property allows adjusting a minimum variation level of an OPC Tag, so that it can be updated. This property only applies to OPC Group Tags that are considered as of analog type by the OPC Server to which the OPC Driver is connected. The valid interval for this property is between 0 and 100%. A value of 0 (zero) for this property means that any variation in the value of an OPC Group Tag implies in updating this OPC Group. This percentage applies to each Tag according to their engineering limits (defined in the OPC server). To update a Tag, the following expression must be true (evaluated in the OPC server):

```
Abs(Previously_stored_value - Current_value) > (DeadBand / 100)
* Abs(Upper_limit - Lower_limit)
```

The default value of this property is 0 (zero).

5.5.2.3.2.3 Enable

◩ This property enables updating Tags inside an OPC Group. If this property is False, no Tag inside the OPC Group is updated. Otherwise, Tags with their **AllowRead** property set to True and in **Advise** mode (for more information, please check the **AdviseType** property) are kept updated according to the update time (the **Scan** property) and OPC Group's dead band (the **DeadBand** property). If this property is False, it is not possible to use the OPC Group's **Refresh** method.

5.5.2.3.2.4 RealScan

9 Scan time effectively used by the OPC server.

5.5.2.3.2.5 Scan

9 Specifies the update's scan time of OPC Group Tags that is used by the server. This property is represented in milliseconds and it can be modified after starting communication, and it is used only when the **Enable** property is set to True.

When configuring this property in several Tags in the application, users should increase the value of this property on those OPC Group Tags that do not vary very much in the device, enabling other OPC Group Tags with higher priority to be read more frequently, thus enhancing application's performance and responsiveness. The default value of this property is 1000.

5.5.2.3.3 OPC Block

This section contains information about **events**, **methods**, and **properties** of the OPC Block object.

5.5.2.3.3.1 Events

This section contains information about the events of the OPC Block object.

OnRead

OnRead()

This event occurs when an OPC Block value is received from an OPC Server. Use the **OnRead** event when there is a need to perform an operation right after modifying data in the OPC Block (the **Bit00 to Bit31**, **Quality**, **RawValue**, **TimeStamp**, and **Value** properties of any OPC Block Element).

5.5.2.3.3.2 Methods

This section contains information about the methods of the OPC Block object.

Write

Write()

Writes the OPC Block's current value to the device. For more information, please check the Driver's documentation. This method returns a **Boolean** indicating whether this operation succeeded or not.

WriteEx

WriteEx([*Value*[, *SyncWrite*]])

Writes a value to the device. All its parameters are optional. If omitted, the behavior of this method is the same as the **Write** method. This method returns a **Boolean** indicating whether this operation succeeded or not. The *Value* parameter defines the value to be written to the OPC Driver. The data type depends on the Driver. If omitted, assumes the OPC Block's current value. The *SyncWrite* parameter is a **Boolean** specifying whether this operation must be synchronous (True) or asynchronous (False). If omitted, uses the value specified in the OPC Block's **SyncWrite** property.

NOTE: As in the **Write** method, writings are performed independent of the value being different from the OPC Block's current value, as well as independent of the OPC Block's **AllowWrite** property being True or False. In addition, if the writing works but the OPC Block is not in scan (whether its **AllowRead** property is set to False or because it is using the **AdviseWhenLinked** option when it is not linked), the written value immediately assumes a good quality and the time stamp of the moment of writing.

5.5.2.3.3.3 Properties

This section contains information about the properties of the OPC Block object.

AdviseType

Controls the **Advise** mode. The available options are described on the next table.

Available options for the AdviseType property

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The Tag is updated if the OPC Block's **AllowRead** property is True and the OPC Group's **Enable** property is also True.
1 - AdviseWhenLinked	The Tag is updated only if the OPC Block's **AllowRead** property and the OPC Group's Enable property are True, and the Tag is associated to an active object, such as a Display on an open Screen or an enabled Alarm, among others. A Tag link for this purpose can be assigned to the following properties: **Value**, **RawValue**, **Quality**, and from **Bit00 to Bit31** of OPC Block Elements, and **Quality** and **TimeStamp** of OPC Blocks.

Example:

```
Sub CommandButton3_Click()
  MsgBox Application._
    GetObject("OPCDriver.OPCGroup.SCRIPT1").AdviseType
End Sub
```

AllowRead

◩ Configure this property to define whether this OPC Block must or must not be read by the OPC Driver. If this property is configured to True, then the Driver automatically updates OPC Block Element's **Value** and Bits (from **Bit00 to Bit31**) properties in time spans. Otherwise, this OPC Block is not read if this property is set to False. This property can be modified at run time. Its default value is True. Example:

```
Sub Button1_Click()
  ' Stops tag reading
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowRead = False
End Sub
```

AllowWrite

◩ Configure this property to define whether this Tag must be automatically written when the **Value** property or any Bit property (from **Bit00 to Bit31**) is modified. If this property is True, changes are sent to the device or equipment linked to the OPC Driver, otherwise these changes are ignored. If this property is configured to True, then the Driver automatically updates **Value** and Bit (from **Bit00 to Bit31**) properties of this object, in time spans. Otherwise, this OPC Block is not read. The default value of this property is True. Example:

```
Sub Button1_Click()
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowWrite = False
End Sub
```

DataType

Read-only property. Determines the data type associated to this OPC Block, according to the next table.

Available options for the DataType property

OPTION	DESCRIPTION
0 - _Undefined	Unidimensional undefined value (Empty).
1 - _Null	Null value.
2 - _Integer	Unidimensional 16-bit signed integer value.
3 - _Long	Unidimensional 32-bit signed integer value.
4 - _Single	Unidimensional 32-bit floating point value.
5 - _Double	Unidimensional 64-bit floating point value.
6 - _Currency	Unidimensional currency value with four decimal places.
7 - _Date	Date and time value.
8 - _String	Literal value (text).

OPTION	DESCRIPTION
9 - _Object	Unidimensional reference value to an object.
10 - _Error	Unidimensional error code value.
11 - _Boolean	Unidimensional **Boolean** value (true or false).
12 - _Variant	Data of any type used for objects and other values to which the data type is unknown.
13 - _UnkObject	Unidimensional reference value to an object.
14 - _Decimal	Unidimensional 96-bit floating point value.
36 - _Record	Unidimensional recording value.
16 - _Char	Unidimensional 8-bit integer value.
17 - _Byte	Used to create DLLs and OLE, uses one byte in memory.
18 - _Word	Unidimensional 16-bit integer value.
19 - _Dword	Unidimensional 32-bit integer value.
20 - _LongLong	Unidimensional 64-bit signed integer value.
21 - _DDWord	Unidimensional 64-bit integer value.
22 - _Integer_	Integer number, ranges from -32.768 to 32.767, uses two bytes.
23 - _Uinteger	Unsigned integer value (equivalent to a **DWord**), ranges from 0 (zero) to 4294967295 (2^{32} - 1).
8194 - _ArrInteger	Unidimensional array of integer values.
8195 - _ArrLong	Unidimensional array of 32-bit signed integer values.
8196 - _ArrSingle	Unidimensional array of 32-bit floating point values.
8197 - _ArrDouble	Unidimensional array of 64-bit floating point values.
8198 - _ArrCurrency	Unidimensional array of currency values with four decimal places.
8199 - _ArrDate	Unidimensional array of date and time values.
8200 - _ArrString	Unidimensional array of literal values (text).
8201 - _ArrObject	Unidimensional array of reference values to an object.
8202 - _ArrError	Unidimensional array of error code values.
8203 - _ArrBoolean	Unidimensional array of **Boolean** values (true or false).
8204 - _ArrVariant	Array of data of any type used for objects and other values to which the data type is unknown.

OPTION	DESCRIPTION
8205 - _ArrUnkObject	Unidimensional array of reference values to an object.
8206 - _ArrDecimal	Unidimensional array of 96-bit floating point values.
8228 - _ArrRecord	Unidimensional array of record values.
8208 - _ArrChar	Unidimensional array of char values.
8209 - _ArrByte	Unidimensional array of bytes, which are values used to create DLLs and OLE, uses one byte in memory.
8210 - _ArrWord	Unidimensional array of 32-bit integer values.
8211 - _ArrDWord	Unidimensional array of 32-bit integer values.
8212 - _ArrLongLong	Unidimensional array of 16-bit integer values.
8213 - _ArrDDWord	Unidimensional array of 32-bit signed integer values.
8214 - _ArrInteger_	Unidimensional array of 16-bit signed integer values.
8215 - _ArrUInteger	Unidimensional array of unsigned integer values (equivalent to a **DWord**), ranges from 0 (zero) to 4294967295 (2^{32} - 1).

EnableDriverEvent

◩ This property is used to control the generation of the **OnTagRead** event, which occurs in the OPC Driver that contains the OPC Block. If the OPC Block's **EnableDriverEvent** property is set to True, at each reading that comes from the OPC server, whether on error or not, the **OnTagRead** event is generated on the OPC Driver that contains this OPC Block. Otherwise, this event does not occur. Also, when the **EnableDriverEvent** property is set to True, at each writing sent to the OPC server, an **OnTagWrite** event is generated in the OPC Driver that contains the OPC Block. If this writing is asynchronous, the **OnTagWrite** event is generated only when the server sends an answer indicating whether the writing succeeded or not. In this case, the event is only generated if the **EnableDriverEvent** property is set to True at that moment, and not at the moment of sending the writing. The default value of this property is False.

ItemID

A This property determines the path identifying the OPC Block on the server to which the OPC Driver connects. Definition of this path is flexible and depends on the specific server. Usually, servers specify an ID space with hierarchical items, such as **ParentItem.ChildItem.Tag1**. The **ItemID** property works as a unique key for data, considering *where* or *what* allows an OPC server to connect to a data source. Although its default value is empty, users must specify a value for this OPC Block to be valid.

Quality

9 This property informs the quality of the value contained in the **Value** property. Each time the OPC Driver attributes a new value to the OPC Block, it also configures the quality of that data. This is a read-only property. The default value of this property is 0 (zero, bad quality).

NOTE: For more information about quality, please check the topic **Quality** on **E3 User's Manual**.

Size

9 Defines the size of OPC Block's set of values. Please check Driver's documentation, for information about the limit for this property, according to *B1* to *B4* parameters. By creating Elements for the OPC Block, it allows both accessing the values read and also allows writing values to the device. This property cannot be modified once communication has started. The default value of this property is 0 (zero). Example:

```
Sub OPCBlock1_OnStartRunning()
  Size = 12
End Sub
```

SyncWrite

◩ This property determines the type of writing used by an OPC Block. If this property is set to True, this writing is synchronous, that is, the OPC Driver waits for the result of a writing from the server. Otherwise, this writing is asynchronous, that is, OPC Block's value is sent and OPC Driver processing continues immediately. The default value of this property is False.

NOTE: In asynchronous mode (property disabled), communication performance tends to be better, but in synchronous mode (property enabled), success of a writing operation is verified and informed.

TimeStamp

This property is updated whenever there is a change in value or status in either **Value** or **Quality** properties. This property informs a date and time linked to OPC Block's value, as well as to OPC Block's quality. This is a read-only property. The default value of this property is "00:00:00".

5.5.2.3.3.4 OPC Block Element

This section contains information about **events** and **properties** of the OPC Block Element object. This object does not contain methods associated to it.

Events

This section contains information about the events of the OPC Block Element object.

OnRead

OnRead()

Occurs when an OPC Block Element's value is received from an OPC Server. Use the **OnRead** event when there is a need to perform an operation right after modifying any data in the OPC Block Element (the **Bit00 to Bit31**, **Quality**, **RawValue**, or **Value** properties).

Properties

This section contains information about the properties of the OPC Block Element object.

Bit00 to Bit31

These properties represent all 32 bits of an OPC Block Element's **Value** property, where **Bit00** is the least significant bit and **Bit31** is the most significant bit. Modify each one of these bits implies in changing the OPC Block Element's **Value** property and vice versa, but this only happens when the **UseBitFields** property is set to True. The default value of these properties is False.

DeviceHigh

This property defines the highest value reached by an OPC Block Element in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

DeviceLow

This property defines the lowest value reached by an OPC Block Element in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default

value of this property is 0 (zero). Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EnableScaling

◩ This property enables or disables the scale of values for a value sent or received from a device. If this property is set to True, all values from the device receive a scale adjustment according to **DeviceHigh**, **DeviceLow**, **EUHigh**, and **EULow** properties before setting it to the **Value** property. The same occurs when a writing is needed, when the value in the **Value** propery receives a scale adjustment (but without changing the **Value** property) and later it is sent to the device. If the **EnableScaling** property is set to False, no scale adjustment is performed in neither way (for writing and for reading). The default value of this property is False. Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EU

A Identifies the engineering unit represented by this value, such as degrees, meters, etc. Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
```

```
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EUHigh

Defines the highest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EULow

Defines the lowest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

Index

Use this property to specify the position that an OPC Block Element occupies among the Elements configured in the OPC Block's **Size** property where it is inserted.

This property's accepts values from 0 (zero) to the value defined by **Size** minus one. For example, configuring an OPC Block Element's **Size** property to 20, the maximum valid number for the **Index** property is 19 and the minimum is 0 (zero). This property can be changed after starting communication. The default value of this property is 0 (zero), but when mapping Elements of an OPC Block, Studio automatically sets the **Index** property to the specified value. Example:

```
Sub Element1_OnRead()
  MsgBox Index
End Sub
```

Quality

This property represents the quality status of a value in the **Value** property.

NOTE: For more information on quality, please check the topic **Quality** on **E3 User's Manual**.

RawValue

It has access to the original value of an OPC Block Element previous to the scale, that is, regardless of **EnableScaling** property's configuration. Therefore, if this property is set to False, the **Value** and **RawValue** properties have the same behavior.

UseBitFields

If this property is set to True, every time the value in the **Value** property changes, it updates all bits referring to the **Bit00 to Bit31** properties. Likewise, it updates the value in the **Value** property every time any of the **Bit00 to Bit31** properties change their values, and later send it to the device if the OPC Block's **AllowWrite** property is set to True. Otherwise, if this property is set to False, bits do not change nor lead to any modification. This property can be updated after starting communication. The default value of this property is False.

Value

Updated whenever a new value is read from the OPC server, according to OPC Block's **ItemID** property specifications where this OPC Block Element is inserted, and also considering the **Index** property, which specifies the position of an Element in an OPC Block array. This property's data type (integer, floating point, text, etc.) depends on the Driver to which it is associated and its configuration.

This property is only updated this way if the **AllowRead** property of the OPC Block to which the OPC Block Element belongs is set to True, and according to a scan time defined in the **Scan** property of the OPC Group that contains the OPC Block. If the OPC Block's **AllowWrite** property is set to True, users can write values to the device simply by attributing a new value to the **Value** property.

This is also the default property of an OPC Block Element. Therefore, a reference by value to an OPC Block Element does not need necessarily to explicit its **Value** property to have access to its value. If this property is not being updated, check if

the Index property is correctly configured (its value must be between zero and the size of the OPC Block minus one). Example:

```
Sub Button1_Click()
  ' Accesses an Element and shows its current value
  ' elm1 is an OPC Block Element object
  Set obj = Application.GetObject_
    ("OPCDriver1.Group1.OPCBlock1.elm1")
  MsgBox "Elm1's current value: " & obj.Value
  ' This can also be performed in a different way,
  ' without displaying the Value property, which is default
  MsgBox "Elm1's current value: " & obj
End Sub
```

5.5.2.3.4 OPC Tag

This section contains information about **events**, **methods**, and **properties** of the OPC Tag object.

5.5.2.3.4.1 Events

This section contains information about the events of the OPC Tag object.

OnRead

OnRead()

Occurs when an OPC Tag's value is received from an OPC Server. Use the **OnRead** event when there is a need to perform an operation right after modifying any data in the OPC Tag (the **Bit00 to Bit31**, **Quality**, **RawValue**, **TimeStamp**, or **Value** properties).

5.5.2.3.4.2 Methods

This section contains information about the methods of the OPC Tag object.

Write

Write()

Performs a writing of OPC Tag's current value to the device. For more information, please check the Driver's documentation. This method returns a **Boolean** indicating whether this operation succeeded or not.

WriteEx

WriteEx([*Value*[, *SyncWrite*]])

Writes a value to the device. All its parameters are optional. If omitted, the behavior of this method is the same as the **Write** method. This method returns a **Boolean** indicating whether this operation succeeded or not. The *Value* parameter defines the value to be written to the OPC Driver. The data type depends on the Driver. If omitted, assumes the OPC Tag's current value. The *SyncWrite* parameter is a **Boolean**

specifying whether this operation must be synchronous (True) or asynchronous (False). If omitted, uses the value specified in the OPC Tag's **SyncWrite** property.

NOTE: As in the **Write** method, writings are performed independent of the value being different from the OPC Tag's current value, as well as independent of the OPC Tag's **AllowWrite** property being True or False. In addition, if the writing works but the OPC Tag is not in scan (if its **AllowRead** property is set to False or because it is using the **AdviseWhenLinked** option when it is not linked), the written value immediately assumes a good quality and the time stamp of the moment of the writing.

5.5.2.3.4.3 Properties

This section contains information about the properties of the OPC Tag object.

AdviseType

Controls the **Advise** mode. The available options are described on the next table.

Available options for the AdviseType property

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The OPC Tag is kept updated if its **AllowRead** property is set to True, and the OPC Group's **Enable** property is also set to True.
1 - AdviseWhenLinked	The OPC Tag is only updated if the OPC Group's **AllowRead** and **Enabled** property are set to True, and the Tag is linked to an active object, such as a Display on an open Screen or an enabled Alarm, among others. A Tag link for this purpose can be performed in the following OPC Tag properties: **Value**, **RawValue**, **TimeStamp**, **Quality** and **Bit00 to Bit31**.

Example:

```
Sub CommandButton3_Click()
  MsgBox Application._
    GetObject("OPCDriver.OPCGroup.OPCTag1").AdviseType
End Sub
```

AllowRead

Defines whether this OPC Tag should be read or not by the OPC Driver. If this property is set to True, the Driver automatically updates **Value** and Bit (from **Bit00 to Bit31**) properties of this object in time spans. Otherwise, this OPC Tag is not read. This property can be modified at run time. The default value of this property is True. Example:

```
Sub CommandButton1_Click()
  ' Stops a Tag reading
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowRead = False
End Sub
```

AllowWrite

◩ Defines whether this OPC Tag should be written automatically or not when the **Value** or any Bit (from **Bit00 to Bit31**) properties are modified. If this property is set to True, these modifications are sent to the device associated to the OPC Driver. Otherwise, these modifications are ignored. The default value of this property is True. Example:

```
Sub Button1_Click()
  ' Disables a Tag writing
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowWrite = False
End Sub
```

Bit00 to Bit31

◩ These properties represent all 32 bits of an OPC Tag's **Value** property, where **Bit00** is the least significant bit and **Bit31** is the most significant bit. Modify each one of these bits implies in changing the OPC Tag's **Value** property and vice-versa, but this only happens when the **UseBitFields** property is set to True. The default value of this property is False.

NOTE: Bit values (the **Bit00** to **Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

DataType

Read-only property. Determines the data type associated to this OPC Tag. Please check the table **Available options for the DataType property** on OPC Block's **DataType** property for all available values for this property.

DeviceHigh

9 This property defines the highest value reached by this OPC Tag in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
```

```
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

DeviceLow

9 This property defines the lowest value reached by this OPC Tag in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

EnableDriverEvent

◩ This property is used to control the generation of the **OnTagRead** event, which occurs in the OPC Driver that contains the OPC Tag. If the Tag's **EnableDriverEvent** property is set to True, at each reading that comes from the OPC Server, whether on error or not, the **OnTagRead** event is generated on the OPC Driver that contains this Tag. Otherwise, this event does not occur. Also, when the **EnableDriverEvent** property is set to True, at each writing sent to the OPC Server, an **OnTagWrite** event is generated in the OPC Driver that contains the OPC Tag. If this writing is asynchronous, the **OnTagWrite** event is generated only when the server sends an answer indicating whether the writing succeeded or not. In this case, the event is only generated if the **EnableDriverEvent** property is set to True at that moment, and not at the moment of sending the writing. The default value of this property is False.

EnableScaling

◩ If this property is set to True, all values from the device receive a scale adjustment, according to **EUHigh** and **EULow** properties before setting it to the **Value** property. If this property is set to False, no scale adjustment is performed in neither way (for writing and for reading). The default value of this property is False. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EU

A Identifies the engineering unit represented by the value, such as degrees, meters, etc. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EUHigh

9 Defines the highest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
```

```
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

EULow

9 Defines the lowest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

ItemID

A This property determines the path that identifies the OPC Tag on the server to which the OPC Driver connects. Definition of this path is flexible and depends on the specific server. Usually, servers specify an ID space with hierarchical items, such as **ParentItem.ChildItem.Tag1**. The **ItemID** property works as a unique key for data, considering *where* or *what* allows an OPC server to connect to a data source. Although its default value is empty **String**, users must specify a value for this OPC Tag to be valid.

Quality

9 This property informs the quality of the value contained in the **Value** property. Each time the OPC Driver attributes a new value to the OPC Tag, it also configures

the quality of that data. This is a read-only property. The default value of this property is 0 (zero, bad quality).

NOTE: For more information about quality, please check the topic **Quality** on **E3 User's Manual**.

RawValue

It has access to the original value of the OPC Tag previous to the scale, that is, regardless of **EnableScaling** property's configuration. Therefore, if this property is set to False, the **Value** and **RawValue** properties have the same behavior.

SyncWrite

This property determines the type of writing used by an OPC Tag. If this property is set to True, this writing is synchronous, that is, the OPC Driver waits for the result of a writing from the server. Otherwise, this writing is asynchronous, that is, OPC Tag's value is sent and OPC Driver processing continues immediately. The default value of this property is False.

NOTE: In asynchronous mode (property set to False), communication performance tends to be better, but in synchronous mode (property set to True), success of a writing operation is verified and informed.

TimeStamp

UThis property is updated whenever there is a change in value or status in either **Value** or **Quality** properties. This property informs a date and time linked to OPC Tag's value, as well as to OPC Tag's quality. This is a read-only property. The default value of this property is "00:00:00".

UseBitFields

If this property is set to True, every time the value in the **Value** property changes, it updates all bits referring to the **Bit00 to Bit31** properties. Likewise, it updates the value in the Value property every time any of the **Bit00 to Bit31** properties change their values, and later send it to the device if the OPC Tag's **AllowWrite** property is set to True. Otherwise, if this property is set to False, bits do not change nor lead to any modification. This property can be updated after starting communication. The default value of this property is False.

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

Value

This property is updated whenever performing a new valid reading of a value from a device using its configuration, but the data type of this variable (integer, floating point, or text) depends on the OPC Driver to which it is linked and on its

configuration. This property is only updated this way if the **AllowRead** property is set to True and there is no communication errors (in this case only the **Quality** and **TimeStamp** properties are updated), but according to a scan time defined on the OPC Group to which the OPC Tag belongs. Another way of using this property is writing values to the device. JUst set a new value to the **Value** property or any of the **Bit00 to Bit31** bits, as long as the **AllowWrite** property is set to True. This property is also the standard property of an OPC Tag object. Thus, a reference by value to an OPC Tag object does not need to explicit use the **Value** property to access its value. The default value of this property is empty. Example:

```
Sub Button1_Click()
  ' Accesses a Tag and shows its current value
  ' tag1 is an OPCTag-type object
  Set obj = Application._
    GetObject("CommDriver1.tag1")
  MsgBox "Tag1's current value: " & obj.Value
  ' Without showing the Value property,
   which is the default one
  MsgBox "Tag1's current value: " & obj
End Sub
```

5.5.3 OPC UA Driver

This section contains information about **properties** of the OPC UA Driver object. This object does not contain events nor methods associated to it.

5.5.3.1 Properties

This section contains information about the properties of the OPC UA Driver object.

5.5.3.1.1 EndPointURL

A Read and write property that specifies the path (endpoint) of the OPC UA server to which the client connects. This property cannot be changed while the connection is active.

5.5.3.1.2 Password

A Read an write property that specifies the user's password used in the connection with an OPC UA server. This property is used together with the **UserName** property and its default value is an empty **String**.

NOTE: This property can be changed while the communication is active, but that change is only effective when the OPC UA Client is restarted.

5.5.3.1.3 SecurityMode

Read and write property that specifies the security mode used when connecting to an OPC UA server. This property cannot be changed while the connection is active. Possible values for this property are the following:

- **1**: usmNone (does not use any security mode in the connection)
- **2**: usmSign (uses authentication in the connection)
- **3**: usmSignAndEncrypt (uses authentication and encryption in the connection)

This property is used together with the **SecurityPolicy** property to determine the type of security of the connection. The default value of this property is 1 (**usmNone**), that is, no security.

NOTE: If the **SecurityMode** property is different from **usmNone**, the OPC UA server must accept the certificate from E3 client's instance. The way a certificate is accepted or recognized by an OPC UA server depends on the server.

5.5.3.1.4 SecurityPolicy

Read and write property that specifies the security policy (encription) used when connecting to an OPC UA server. This property cannot be changed while the communication is active. Possible values for this property are the following:

- **1**: uspNone (does not use encryption in the connection)
- **2**: uspBasic128Rsa15 (uses the RSA algorithm with a 128-bit key in the connection)
- **3**: uspBasic256 (uses the AES algorithm with a 256-bit key in the connection)

This property is used together with the **SecurityMode** property to determine the type of security of the connection. The default value of this property is 1 (**uspNone**), that is, no security policy.

5.5.3.1.5 TimeoutCall

9 Read and write property that specifies a time limit for a call to an OPC UA server, in milliseconds. This property cannot be changed while the communication is active, its value must be greater than 0 (zero), and its default value is 10000 (10 seconds).

NOTE: In the OPC UA standard, several calls allow to continue the communication, with multiple responses from the server. So, this time ends up being the limit time for a response from the OPC UA server.

5.5.3.1.6 TimeoutConnection

9 Read and write property that specifies the time limit of a connection, in milliseconds. The default value of this property is 10000 (10 seconds) and its value must be greater than 0 (zero). This property cannot be changed while the communication is active.

5.5.3.1.7 TimeoutSession

9 Read and write property that specifies a time limit for renewing a communication session from an E3 client to an OPC UA server, in milliseconds. The default value of this property is 600000 (10 minutes) and this value must be greater than 0 (zero). This property cannot be changed while the communication is active.

5.5.3.1.8 UserName

A Read and write property that specifies the user's name used in the OPC UA server's connection. This property is used together with the **Password** property. The default value of this property is an empty **String**.

NOTE: This property can be changed while the communication is active, but that change is only effective when the OPC UA Client is restarted.

5.6 Data Server

This section contains information about events, methods, and properties of the following objects: **Query**, **Data Folder**, **Counter Tag**, **Demo Tag**, **Internal Tag**, and **Timer Tag**.

5.6.1 Query

This section contains information about **events**, **methods**, and **properties** of the Query object.

5.6.1.1 Events

This section contains information about the events of the Query object.

5.6.1.1.1 OnAsyncQueryFinish

OnAsyncQueryFinish(*Recordset*, *Error*)

Occurs when the **GetAsyncADORecordset** method returns. The *Recordset* parameter is the ADO Recordset object generated by the Query, and the *Error* parameter is a **Boolean** that, when True, indicates that the object could not be generated. Example:

```
Sub query1_OnAsyncQueryFinish(Recordset, Error)
```

```
  MsgBox "Returned " + CStr(Recordset.RecordCount) + " records"
End Sub
```

5.6.1.2 Methods

This section contains information about the methods of the Query object.

5.6.1.2.1 AddField

AddField(*Name*[, *Table*])

The **AddField** method adds a new field from the table in the Query. The *Name* parameter determines the name of the new field to add to the Query. The *Table* parameter represents the name of the table to which the field belongs. This method was developed only for compatibility with E3Chart's old Query object. Example:

```
Sub Button1_Click()
  Screen.Item("E3Browser").Item("Query").AddField "Field1"
End Sub
```

NOTE: This method will fail if the **QueryType** property is configured to **1 - qtStorage**.

5.6.1.2.2 AddStorageTag

AddStorageTag(*Name*, *FieldType*)

Adds a Tag from the Storage to the Query. The *Name* parameter receives the name of the Tag to add. The *FieldType* parameter indicates the Tag type (**0: Double**, **1: Bit**, **2: String**, **3: Integer**). Returns a **Boolean** indicating whether the operation was successful or not.

NOTE: This method will fail if the **QueryType** property is configured to **0 - qtDBServer**.

5.6.1.2.3 AddTable

AddTable(*Name*)

Adds a table from the database to the Query. The *Name* parameter determines the name of the table to add.

NOTE: This method will fail if the **QueryType** property is configured to **1 - qtStorage**.

5.6.1.2.4 AddUaField

AddUaField(*Name*[, *Alias*[, *Function*]])

This method adds, at run time, a field to a query configured as an **OPC UA**-type. The methods returns True if successful and False on failure. The parameters of this method are the following:

- **Name**: A **String** with the path (Column) of the field
- **Alias**: A **String** with the title of the field. If this parameter is omitted, assumes an empty **String**
- **Function**: A **String** with the type of aggregation function of the field, in case a **Processed Data**-type query is used. If this parameter is omitted, assumes the value **Interpolative**

In a **Raw Data**-type query (the **UaQueryType** property equal to zero), this method fails if there is already a field defined. This method is only effective if the **QueryType** property is configured as **OPC UA** (value equal to **2: qtOpcUa**). Otherwise, it returns False.

5.6.1.2.5 Execute

Execute(*ImmediateExecute*)

The **Execute** method executes a SQL command that does not have a return (such as **DELETE**, **UPDATE**, or **INSERT**), configured in the Query's **SQL** property. The *ImmediateExecute* parameter indicates whether the operation passes through database operation queues (.e3i and .e3o files) before it reaches the database (if False), or it is sent directly to the database (if True). The advantage of using a Query to execute commands is using variables, such as in a simple query.

Example of SQL commands:

```
DELETE FROM test WHERE cod > 10
UPDATE test SET cod = 10 WHERE cod > 10
INSERT INTO test (cod) VALUES(10)
```

Example:

```
Sub CommandButton1_Click()
  Screen.Item("Query1").Execute
End Sub
```

5.6.1.2.6 GetADORecordSet

GetADORecordSet()

The **GetADORecordSet** method returns an **ADO**-type (*ActiveX Data Object*) Recordset as the result of executing the configured Query. Example:

```
Sub Button1_Click()
  Set rec = Screen.Item("Query1").GetADORecordset()
  strDates = " "
  i = 0

  ' Displays a message with the first 10 records
  ' of the E3TimeStamp column
  While (NOT rec.EOF AND i < 10)
```

```
    strDates = strDates & CStr(rec("E3TimeStamp")) & _
      Chr(10) & Chr(13)
    i = i + 1
    rec.MoveNext()
  Wend
  MsgBox strDates
End Sub
```

NOTE: For more information about the ADORecordset object returned by this method, please check Microsoft documentation at *http://msdn.microsoft.com/en-us/library/ms675841(VS.85).aspx*.

5.6.1.2.7 GetAsyncADORecordSet

GetAsyncADORecordSet()

Creates a Query and, when it finishes, generates the object's **OnAsyncQueryFinish** event, passing the recordset generated by the Query to this event.

5.6.1.2.8 GetE3QueryFields

GetE3QueryFields()

The **GetE3QueryFields** method returns a Fields Collection (columns) of a Query. Every item on that Collection has properties that can be modified, as described on topic **Query Field**. Example:

```
Sub Button1_Click()
  ' Traverses the Fields Collection,
  ' displaying the fields on a message box,
  ' and also setting them as visible
  ' on Query's configuration

  Set Browser = Screen.Item("E3Browser")
  Set Query = Browser.Item("Query")
  Set Fields = Query.GetE3QueryFields()
  For Each field In Fields
    MsgBox CStr(field.TableName) & "-" & CStr(field.ColumnName)
    field.Visible = True
  Next
  ' Performs E3Browser's Query again, which has been
  ' just modified, so that all fields appear.
  Browser.RetrieveE3QueryFields()
  Browser.Requery()
End Sub
```

NOTE: To use this method, the Query must be previously created at design time.

5.6.1.2.9 RemoveField

RemoveField(*FieldName*[, *Table*])

The **RemoveField** method removes a field previously included in a Query. The *FieldName* parameter determines the name of the field to be removed from the Query. The *Table* parameter represents the name of the table to which the field belongs. This method was developed just to keep compatibility with E3Chart's old Query object. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Browser").Item("Query").RemoveField "Field1"
End Sub
```

5.6.1.2.10 RemoveStorageTag

RemoveStorageTag(*Name*)

Removes a previously-configured Tag in the Query. The *Name* parameter indicates the Tag's name. Returns a **Boolean** indicating whether the operation was successful or not.

NOTE: This method will fail if the **QueryType** property is configured to **0 - qtDBServer**.

5.6.1.2.11 RemoveTable

RemoveTable(*TableName*)

Removes a table from the Query. The *TableName* parameter determines the name of the table to remove.

NOTE: This method will fail if the **QueryType** property is configured to **1 - qtStorage**.

5.6.1.2.12 RemoveUaField

RemoveUaField(*Name*)

This method removes the field with the specified name, if it exists. The *Name* parameter is a **String** with the name (Column) of the field. The method returns True if it is successful and False on failure. This method is only effective if the **QueryType** property is configured as **OPC UA** (value equal to **2: qtOpcUa**). Otherwise, it returns False.

5.6.1.2.13 SetVariableValue

SetVariableValue(*VarName*, *Value*)

The **SetVariableValue** method adjusts the value of a variable set in the Query, so

that this value can be informed as a filter or parameter before being performed. The variable name (*VarName*) and the value (*Value*), which can be a number, a text or a date and time must be defined. Example:

```
Sub CommandButton_Click()
  Set cons = Screen.Item("E3Browser1").Item("Query1")
  InitialDate = now - 1
  FinalDate = now
  cons.SetVariableValue "IniDate", InitialDate
  cons.SetVariableValue "EndDate", FinalDate
End Sub
```

5.6.1.3 Properties

This section contains information about the properties of the Query object.

NOTE: It is not advisable accessing these properties directly via scripts. Users should access the Query object, passing parameters through the **SetVariableValue** method, and then modifying filters or fields through the collection returned by the **GetE3QueryFields** method.

5.6.1.3.1 CursorLocation

Defines where the Query is generated and handled, from a DBMS (Database Management System) point of view. The available options are:

- **0 - clServer**: the Query is generated on the DBMS (server).
- **1 - clClient**: the Query is generated on the Server (client).

The default value of this property is **1 - clClient**. See also the **CursorType** property.

NOTE: This property has no effect for Databases with the **SourceType** property equal to **0 - stAccess**, because in this case the Server always generates and handles the Query. However, for **SourceType** equal to **1 - stOracle** or **2 - stSqlServer**, it is advisable to use the option **1 - clClient**. For more information, check the documentation of the **CursorLocation** property and the **CursorLocationEnum** enumeration of the ADO (*ActiveX Data Object*) object.

5.6.1.3.2 CursorType

Defines the type of a Query, according to the way data is viewed. The available options are:

Available options for CursorType

OPTION	DESCRIPTION
0 - ctKeyset	Any change in records initially returned by the Query will be visible (Default).

OPTION	DESCRIPTION
1 - ctStatic	No change in records initially returned by the Query will be visible.
2 - ctDynamic	All new records added to the Query are visible, in addition to the changes in records initially returned by the Query.

5.6.1.3.3 DataSource

A Indicates the Database, Storage, or OPC UA Driver object that is used in the Query. This is a read-only property, but it can be modified at run time.

5.6.1.3.4 Fields

A Text with the fields to be viewed in the Query, separated by commas. It corresponds to the argument of the **SELECT** clause in Query's SQL code. When blank, it determines that the Query must display all table fields. This is a read-only property, but it can be modified at run time.

5.6.1.3.5 FunctionSubType

Specifies the function subtype indicated by the **FunctionType** property. Only the options **1 - ftArchivedValue, 2 - ftTagAttribute**, and **6 - ftCalculatedData** have subtypes. For the other functions, **FunctionSubType** assumes **-1 - fsNoSubType**. The following table shows the possible property values, according to the chosen function in the **FunctionType** property:

Subtypes for ArchivedValue function (FunctionType = 1)

SUBTYPE	DESCRIPTION
0 - fsPreviousArchivedValue	Value stored right before the informed time stamp.
1 - fsInterpolatedArchivedValue	Calculated value, based on previous and next values.
2 - fsNextArchivedValue	Value stored right after the informed time stamp.
3 - fsExactArchivedValue	If a value is found at the exact informed time stamp.

Subtypes for TagAttribute function (FunctionType = 2)

SUBTYPE	DESCRIPTION
0 - fsTagAttributeDescription	Tag's meaning or description.
1 - fsTagAttributeSource	Path of the Tag being stored.
2 - fsTagAttributeType	Data type: **Double**, **Boolean**, or **String**.
3 - fsTagAttributeEU	Engineering unit.
4 - fsTagAttributeLowEng	Lower limit.

SUBTYPE	DESCRIPTION
5 - fsTagAttributeHighEng	Upper limit.
6 - fsTagAttributeDeadBand	Dead band for recording.
7 - fsTagAttributeDeadBandUnit	Dead band unit (an absolute value or a percentage).
8 - fsTagAttributeMinRecTime	Minimum recording time (variations smaller than this interval are not recorded).
9 - fsTagAttributeMaxRecTime	Maximum recording time (the absence of variation on these intervals forces a recording).

Subtypes for CalculatedData function (FunctionType = 6)

SUBTYPE	DESCRIPTION
0 - fsTotalCalculatedData	Total of values.
1 - fsMinimumCalculatedData	Minimum value.
2 - fsMaximumCalculatedData	Maximum value.
3 - fsStandardCalculatedData	Standard deviation.
4 - fsRangeCalculatedData	Range of values.
5 - fsMeanCalculatedData	Mean of values.
6 - fsMedianCalculatedData	Median of values.

5.6.1.3.6 FunctionType

This property is valid when a Storage object is the Query's source (this is indicated by the **DataSource** property). It specifies the function that will define data generated by the Query. Some functions have subfunctions, which can be indicated in the **FunctionSubType** property. The **FunctionType** property can assume the following values:

Available options for FunctionType

OPTION	DESCRIPTION
-1 - ftNoFunction	There is no defined function. This value is not available when the **QueryType** property is configured to **1 - qtStorage**.
0 - ftLastValue	Returns the last value stored in the Database.
1 - ftArchivedValue	Returns a stored value relative to a predefined time stamp defined in the `TimeStamp` variable. The relationship is specified in the **FunctionSubType** property.
2 - ftTagAttribute	Returns a Tag attribute, defined in the **FunctionSubType** property.

OPTION	DESCRIPTION
3 - ftCompressedDataNValues	Returns, for a single Tag, *N* values defined in the variable NumVals, stored from an initial time, defined in the variable StartTime.
4 - ftCompressedDataStartEndTime	Returns, for a single Tag, the values stored at the interval between the variables StartTime and EndTime.
5 - ftSampledData	Returns, for one or more Tags, the interpolated values (that is, estimated) at the interval between the variables StartTime and EndTime, in fixed intervals defined by the variable TimeInterval.
6 - ftCalculatedData	Returns, for one or more Tags, the result of the math operations applied to data at the interval between variables StartTime and EndTime, in fixed intervals defined by the variable TimeInterval.

NOTE: Variables can also be defined at run time using the Query's **SetVariableValue** method.

5.6.1.3.7 GroupBy

A Text corresponding to the argument of the **GROUP BY** clause in Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.3.8 Having

A Text corresponding to the argument of the **HAVING** clause in Query's SQL code. This property is commonly used with the **GroupBy** property. This is a read-only property, but it can be modified at run time.

5.6.1.3.9 IgnoreQuality

◩ Allows indicating whether bad quality data is included or not on Query's results. This property is only effective if the Query object is using a Storage as its data source. This property can be changed by script at run time.

5.6.1.3.10 OrderBy

A Text corresponding to the argument of the **ORDER BY** clause in Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.3.11 QueryType

Indicates the type of query to perform. Possible values for this property are the following:

- **-1 - qtUndefined**: The Query object initially tries to execute the query on a Database object. If it fails, then it tries to execute the query on a Storage object. This is the default value for Queries created in Studio or at run time
- **0 - qtDBServer**: The Query object tries to execute the query only on a Database object
- **1 - qtStorage**: The Query object tries to execute the query only on a Storage object
- **2 - qtOpcUa**: The Query object tries to execute the query only on an OPC UA Driver object

NOTES:

- In case of manipulating Queries at run time, it is recommended to configure this property with the desired type before performing other settings on the object, specially when reusing a Query object for different data sources.
- For applications created on previous versions, when opening the application in Studio this property is automatically configured to **-1 - qtUndefined**.
- The **AddStorageTag** and **RemoveStorageTag** methods will fail if this property is configured to **0 - qtDBServer**.
- The **FunctionType** property does not accept the value **-1 - ftNoFunction** if this property is configured to **1 - qtStorage**.
- The **AddField**, **AddTable**, and **RemoveTable** methods will fail if this property is configured to **1 - qtStorage**.
- If this property is configured to **-1 - qtUndefined**, an attempt to execute the Query on a Storage object (after failing on a Database object) will fail if the **FunctionType** property is configured to **-1 - ftNoFunction** or if the **FunctionSubType** property is configured to an invalid value.

5.6.1.3.12 SQL

Contains the SQL code specified for the Query. This is a read-only property, but it can be modified at run time.

5.6.1.3.13 Table

Contains the tables to be queried (for example, Alarms is the alarms or events table). It corresponds to the argument of the **FROM** clause in Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.3.14 UaNamespaceArray

This property returns an object that is a **Collection of OPC UA Namespaces**, used by the fields configured in the Query. This is a read-only property.

5.6.1.3.15 UaQueryType

This is a read and write property that allows determining whether the Query is a **Raw Data** (**0: uqtRaw**) or **Processed Data** (**1: uqtProcessed**, default value) type. In Studio this property is read-only. At run time it allows configuring the type of OPC UA query, but it only accepts writings if the **QueryType** property is configured as **OPC UA** (value equal to **2: qtOpcUa**).

5.6.1.3.16 Where

Determines the Query condition that filters the table records to be viewed, that is, only the records that meet this conditions are viewed. It corresponds to the **WHERE** argument in the Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.4 Fields Collection

This section contains information about **methods** and **properties** of the Query's Fields Collection object. This object does not have specific events.

5.6.1.4.1 Methods

This section contains information about the methods of the Query's Fields Collection object.

5.6.1.4.1.1 Add

Add(*NewItem*)

Adds a new Query Field object on a Query's Fields Collection, indicated by the *NewItem* parameter.

5.6.1.4.1.2 Item

Item(*Index*)

Returns a Query Field object from the Fields Collection. The *Index* parameter can be the numerical index of this Field, or its name.

5.6.1.4.1.3 RefreshUaNodeIds

RefreshUaNodeIds()

Updates all Query Fields, retrieving the OPC UA Node Identifiers (NodeIds) on the server and updating the **UaNodeId** property of the Query Fields. For this method to work, the following conditions must be met:

- The Query's **QueryType** property must be configured as **2: qtOpcUa**
- The Query must point to a valid OPC UA Driver
- The OPC UA Driver configured in the Query must be active and connected

5.6.1.4.1.4 Remove

Remove(*Index*)

Removes a Query Field object from the Query's Fields Collection. The *Index* parameter can be a **String** with the name of the Query Field, or the index of that object on the Fields Collection.

5.6.1.4.2 Properties

This section contains information about the properties of the Fields Collection object of a Query.

5.6.1.4.2.1 Count

9 Returns the amount of Query Field objects in this Collection. This is a read-only property.

5.6.1.4.3 Query Field

This section contains information about **properties** of the Query Field object. This object does not have specific events nor methods.

5.6.1.4.3.1 Properties

This section contains information about the properties of the Query Field object.

Alias

A Alias of this Field in the Query.

ColumnName

A Column's name. This name must exist on the tables added to the Query.

Criteria

A Filter that will be applied to this Field.

Function

A Function to which this Field can be passed as its parameter.

GroupBy

◩ When in True, indicates that this Field is part of a group.

OrderBy

A Sort order of Field's data. Valid values are "ASC" (ascending), "DESC" (descending), or an empty **String** (no sort order). Any other value means that this Field has no sort order.

OrderNumber

9 Order number of this Field relative to the other Fields that compose the Query's sort order. This value will only be accepted as a number greater than 0 (zero) if the Field has a sort type (defined by the **OrderBy** property). This value must be less or equal to the number of Fields that compose the Query's sort order.

TableName

A Name of the Field's table. This table must have been added during Query's configuration.

UaNodeId

Returns an **OPC UA Node Identifier**-type object, with the configuration of the Node that identifies a field from an OPC UA query on a certain OPC UA server. In Studio this property is only represented by an icon, on the **Id** column of the Query's field configuration.

Visible

◩ When in True, indicates that this Field is visible.

5.6.1.5 OPC UA Namespaces Collection

This section contains information about **methods** and **properties** of the OPC UA Namespaces Collection object (UaNamespaceArray). This object does not have events associated to it.

5.6.1.5.1 Methods

This section contains information about the methods of the OPC UA Namespaces Collection object.

5.6.1.5.1.1 Add

Add(*Item*)

Adds a Namespace identifier, always at the end of the Collection. The *Item* parameter is a **String** that defines the Namespace. This parameter cannot be empty.

5.6.1.5.1.2 Item

Item(*Index*)

Returns the Namespace identifier from the specified index. The *Index* parameter must be a **LONG** ranging between 0 (zero) and **Count** minus 1 (one).

5.6.1.5.1.3 Remove

Remove(*Index*)

Removes the Namespace from the specified index. Notice that this implies in changing the indexes of Namespaces greater than the removed one. It is not possible to remove indexes 0 (zero) and 1 (one). The *Index* parameter is a value (**LONG**) that identifies the Namespace to remove from the Collection, ranging between 2 (two) and **Count** minus 1 (one).

5.6.1.5.2 Properties

This section contains information about the properties of the OPC UA Namespaces Collection object.

5.6.1.5.2.1 Count

9 Returns the total amount of Namespaces in the collection. The minimum value of this property is always 2 (two), as the indexes 0 and 1 are always available.

5.6.1.5.3 OPC UA Node Identifier

This section contains information about **properties** of the OPC UA Node Identifier object (E3UaNodeId). This object does not have events nor methods associated to it.

5.6.1.5.3.1 Properties

This section contains information about the properties of the OPC UA Node Identifier object.

GUID

A Read and write property that identifies this object, in case the **Type** property is equal to **2: nitGUID**. If the **Type** property is different from 2 (two), reading this

property returns an error. Writing to this property always forces the **Type** property to change to 2 (two). The value of this property is a **GUID**-type **String** (*Globally Unique Identifier*), a 128-bit value. In case this property is configured with the value "{00000000-0000-0000-0000-000000000000}", then this object is identified as `Null`.

NamespaceIndex

9 Read and write property that identifies to which Namespace the Identifier refers. This index must be between 0 (zero) and the number of Namespaces minus 1 (one) in the server. When defining a field in a Query, the index refers to the **OPC UA Namespaces Collection** defined in the Query object (which at some point may be different from the server's Namespaces Collection). Its value cannot be greater than 65535, because the OPC UA standard defines this is a 16-bit value.

Numeric

9 Read and write property that identifies this object, in case the **Type** property is equal to **0: nitNumeric**. If the **Type** property is different from 0 (zero), reading this property returns an error. Writing to this property always forces the **Type** property to change to 0 (zero). Setting this property to 0 (zero) identifies this object as `Null`.

Opaque

A Read and write property that identifies this object, in case the **Type** property is equal to **3: nitOpaque**. The value of this property is an array of bytes, that is, a **String** of characters not necessarily valid or printable. If the **Type** property is different from 3 (three), reading this property returns an error. Writing to this property always forces the **Type** property to change to 3 (three). An empty **String** in this property identifies this object as `Null`.

String

A Read and write property that identifies this object, in case the **Type** property is equal to **1: nitString**. If the **Type** property is different from 1 (one), reading this property returns an error. Writing to this property always forces the **Type** property to change to 1 (one). An empty **String** in this property identifies this object as `Null`.

Type

Read and write property that determines the type of identifier this object uses. Possible values for this property are the following:

- **0 - nitNumeric**: The identifier is a number (**LONG**)
- **1 - nitString**: The identifier is a **String**
- **2 - nitGUID**: The identifier is a **GUID**-type (*Globally Unique Identifier*) **String**
- **3 - nitOpaque**: The identifier is an array of bytes

Whenever this property changes, the value of the identifier is forced to `Null`.

5.6.2 Data Folder

The Data Folder object does not have specific events, methods, or properties, only general ones. These can be viewed on section **General Events, Methods, and Properties of Objects**.

5.6.3 Counter Tag

This section contains information about **events** and **properties** of the Counter Tag object. This object does not have methods associated to it.

5.6.3.1 Events

This section contains information about the events of the Counter Tag object.

5.6.3.1.1 OnPreset

OnPreset()

It occurs every time the **Preset** property value is reached.

5.6.3.2 Properties

This section contains information about the properties of the Counter Tag object.

5.6.3.2.1 AutoRestart

◩ Indicates that the counting must restart from zero after it reaches the value established in the **Preset** property. This property is only valid when the **CounterType** property is set to **0 - Preset**.

5.6.3.2.2 CounterType

Defines the counter's behavior. The available values for this property are:

- **0 - Preset**: the counting is stopped when the value set in the **Preset** property is reached
- **1 - Infinite**: the counting goes on indefinitely

5.6.3.2.3 Enabled

◩ Starts or stops the counter. If True, the counting is started, otherwise it is stopped.

5.6.3.2.4 Increment

9 Defines the update interval of the **Value** property. In case the value of this property is modified while the counting is on, this modification will only be effective when the counter is stopped and restarted.

5.6.3.2.5 Preset

9 Limit to be reached by the counter, in seconds. In case the **Preset** value is not a multiple of **Increment**, the Tag reaches this value before the next increment.

5.6.3.2.6 ResetCounterWhenEnabled

◩ Enables restarting the counting (from zero) each time the value of the **Enabled** property returns to True. When disabled, the counter restarts counting from the point it was previously interrupted.

5.6.3.2.7 Value

9 Displays the counting of the counter, in seconds. This is a read-only property. This property gets values multiples of **Increment**, except when the **Enabled** property is set to False. In this case, the value is the one when the Tag was disabled. When restarting the counting, the property value is the next multiple of **Increment**. The maximum value of this property is 2147483647 (0x7FFFFFFF).

5.6.4 Demo Tag

This section contains information about **methods** and **properties** of the Demo Tag object. This object does not have events associated to it.

5.6.4.1 Methods

This section contains information about the methods of the Demo Tag object.

5.6.4.1.1 Reset

Reset()

Resets the phase (movement in time) of Tag's wave format. This phase must not be reset unless the Tag is enabled. This method, when the Tag is enabled, has no effect on **CurrentTime**- and **Random**-type Tags, which are not periodic. When the Tag is disabled, its value is simply reset, regardless of Tag's type. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Data.Sine").Reset()
End Sub
```

5.6.4.2 Properties

This section contains information about the properties of the Demo Tag object.

5.6.4.2.1 Enabled

◩ Defines whether Demo Tag variation is activated. If True, a Demo Tag will update its **Value** property according to the settings in the **Period** and the **Scan** properties. Otherwise, variation is disabled. Default value is True. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Data.DemoTag1").Enabled = True
End Sub
```

5.6.4.2.2 Maximum

9 Determines the maximum Tag value. Default value is 100. Example:

```
Sub CommandButton2_Click()
  ' When clicking the button, a message box is opened,
  ' indicating DemoTag6 Maximum property value
  MsgBox Application.GetObject("Data.DemoTag6").Maximum
End Sub
```

5.6.4.2.3 Minimum

9 Determines the minimum Tag value. Default value is 0. Example:

```
Sub CommandButton2_Click()
  ' When clicking the button, a message box is opened,
  ' indicating DemoTag6 Minimum property value
  MsgBox Application.GetObject("Data.DemoTag6").Minimum
End Sub
```

5.6.4.2.4 Period

9 Defines the wave shape length, in milliseconds. Does not apply when the **Type** property is set to either **0 - Random** or **3 - CurrentTime**. Default value is 10,000 ms. Example:

```
Sub DemoTag1_OnStartRunning()
  Period = 1000
End Sub
```

5.6.4.2.5 Scan

9 Defines the time interval between two variations of the **Value** property, in milliseconds. Use this property to have a greater or smaller amount of data generated by the Demo Tag. Default value is 1,000. Scan value should be greater than 0. Example:

```
Sub Line1_Click()
```

```
  Application.GetObject("Data.DemoTag2").Scan = 200
End Sub
```

5.6.4.2.6 TimeStamp

The **TimeStamp** property is updated whenever a value or state change occurs in the **Value** or **Quality** properties. It informs what is the date and time associated to the Demo Tag value, as well as to its quality. This is a read-only property. Default value is 00:00:00.

5.6.4.2.7 Type

Determines the Tag's wave type. Change this property according to the next table. When the **Type** property is set to **3** (**CurrentTime**), the **Value** property will contain the current server's date and time.

Available options for Type

VALUE	WAVEFORM
0	Random
1	Sine
2	Square
3	CurrentTime
4	RampUp
5	RampDown
6	RampUpDown

Example:

```
Sub Line1_Click()
  Application.GetObject("Data.DemoTag2").Type = 2
End Sub
```

5.6.4.2.8 Value

The **Value** property varies according to the wave form type, established in the **Type** property. This is a read-only property. Default value is 0. Example:

```
Sub CommandButton1_Click()
  MsgBox Application.GetObject("Data.DemoTag2").Value = 10
End Sub
```

5.6.5 Internal Tag

This section contains information about **methods** and **properties** of the Internal Tag object. This object does not have events associated to it.

5.6.5.1 Methods

This section contains information about the methods of the Internal Tag object.

5.6.5.1.1 WriteEx

WriteEx(*NewValue*, *NewTimestamp*, *NewQuality*)

Allows changes to an Internal Tag's value, time stamp, and quality in a single operation. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *NewValue* parameter specifies the Tag's new value; if omitted, Tag's value does not change. The *NewTimestamp* parameter specifies the Tag's new times tamp; if omitted, the time stamp from the moment of method's call is then used. The *NewQuality* parameter specifies Tag's new quality; if omitted, a Good quality (192) is then assumed. All these parameters can be omitted. Example:

```
Sub CommandButton12_Click()
  Dim Ret
  Ret = Application.GetObject("Data.InternalTag1")._
    WriteEx(123.456, "1/1/2001", 193)
  If Ret Then
    MsgBox "It works!"
  Else
    MsgBox "Failed!"
  End If
End Sub
```

5.6.5.2 Properties

This section contains information about the properties of the Internal Tag object.

5.6.5.2.1 Quality

9 Informs the value's quality in the **Value** property. This is a read-write property, but whenever the Internal Tag value is modified, by script or by a Link, it will be updated as well. Example:

```
Sub CommandButton1_Click()
  MsgBox Application.GetObject("Data.InternalTag1").Quality
End Sub
```

NOTE: For more information on quality, see the **Quality** topic on **E3 User's Manual**.

5.6.5.2.2 Retentive

◩ If True, the Internal Tag's value will be stored automatically, in case the active Domain stops. This guarantees that the Tag's value will be synchronized with the

Standby server. Then, when the server starts running, Tag's value will be the same as the server that has stopped. Otherwise, Tag's value will be adjusted to the initial value whenever the Domain has been executed, or the active server switches. This property has no effect if changed at run time. Example:

```
Sub CommandButton1_Click()
  Dim status
  status = Application.GetObject("Data.InternalTag1").Retentive
  MsgBox status
  Select Case status
    Case True
      MsgBox "The internal tag value will be _
        stored automatically."
    Case False
      MsgBox "The tag value will be adjusted to the initial _
        value whenever the Domain is executed or _
        when an active server switch occurs."
  End Select
End Sub
```

NOTE: This property is only valid for Internal Tags contained on a Server. Internal Tags contained on a Viewer cannot be retentive.

5.6.5.2.3 TimeStamp

The **TimeStamp** property informs the date and time associated to the value in the **Value** property. This is a read-write property, but whenever an Internal Tag value is modified, by script or by a Link, it will be updated as well.

5.6.5.2.4 Value

This is a **Variant**-type property, which allows storing values of any type, from an **Integer** to an object pointer (see the next example). Use it to store values in the Viewer or in the Server, and to exchange data among several points in the application. Default value is empty. This is a read-write property. Example:

```
Sub Months_OnStartRunning()
  ' Months is an Internal tag.
  ' The event is used to initialize the array.
  Value = Array("January", "February", "March", _
    "April", "May", "June", "July", "August", _
    "September", "October", "November", "December")
End Sub
```

5.6.6 Timer Tag

This section contains information about **events** and **properties** of the Timer Tag object. This object does not have methods associated to it.

5.6.6.1 Events

This section contains information about the events of the Timer Tag object.

5.6.6.1.1 OnPreset

OnPreset()

It occurs every time the value of **NextExecTime** property is reached.

5.6.6.2 Properties

This section contains information about the properties of the Timer Tag object.

5.6.6.2.1 Enabled

Enables or disables the Timer Tag. The default value of this property is True.

5.6.6.2.2 NextExecTime

Displays the next triggering time. This is a read-only property.

5.6.6.2.3 RepeatInterval

This is used whenever the Tag type is set to **1 - ttContinuous**. The default value of this property is 00:00:01.

5.6.6.2.4 StartTime

Initial Timer Tag's date and time. For the **Single**-type, it is the triggering date and time itself. For other types, it is the moment when the Timer Tag triggers. The default value is the local date and time.

5.6.6.2.5 TriggerType

This is the Timer Tag's trigger type. The available options are:

Available options for TriggerType

OPTION	DESCRIPTION
0 - ttSingle	Single trigger.
1 - ttContinuous	Continuous trigger.
2 - ttDaily	Daily-based trigger.
3 - ttMonthly	Monthly-based trigger.

5.7 Database

This section contains information about **methods** and **properties** of the Database object. This object does not have events associated to it.

5.7.1 Methods

This section contains information about the methods of the Database object.

5.7.1.1 SetDBParameters

SetDBParameters(*ServerName*, *UserName*, *Password*, *DBName*)

A connection **String** with a Database server in the properties of a Database object. The *ServerName* parameter determines a server name. The *UserName* parameter determines a user name. The *Password* parameter determines a password to connect to this database. The *DBName* parameter is the name of a database used in a SQL Server. For other databases, this parameter is not used.

5.7.2 Properties

This section contains information about the properties of the Database object.

5.7.2.1 ConnectionActive

◩ Indicates whether E3 has an active connection with the database. E3 usually uses two connections with this database, one for writing and another one for reading. The **ConnectionActive** property is in True if at least one of those two connections is connected and working. This property should not be used to detect connection failures, as it may be in False in several situations, such as:

- When the database is not in use (no writing or reading operation was executed)
- When the database was just configured (when a connection property of the database changes at run time all connections are closed, and they are reconnected only at the next reading or writing operation)
- When the database connection dropped (for example, a network is not available or the database was closed)
- When the DBServer object is disabled (although a connection, even with a disabled DBServer object, can be reconnected if an application requests an immediate reading or writing to the database)

NOTE: If an application generates database operations consistently, that is, if it is always generating new writings or readings, the **ConnectionActive** property correctly reports the database's connection status, as this eliminates other situations when this property could be in False.

5.7.2.2 EnableSynchronization

Indicates to E3, if enabled (True), that it must also perform a data recording on a second database simultaneously, to improve security. If this property is enabled and there is a Standby server, E3 then performs a syncing between databases from both servers. The default value of this property is False, that is, syncing is disabled.

5.7.2.3 NetLibrary

Configures the type of network library of a Database. The available options are described on the next table.

Available options for NetLibrary property

OPTION	DESCRIPTION
0 - Default	Default library type
1 - NamedPipes	**Named Pipes**-type library
2 - TcpIp	**Winsock TCP/IP**-type library
3 - SpxIpx	**SPX/IPX**-type library
4 - BanyanVines	**Banyan Vines**-type library
5 - MultiProtocol	**Multi-protocolo (RPC)**-type library

NOTE: This **NetLibrary** property is only available on SQL Server databases.

5.7.2.4 nRetries

This property specifies the number of times E3 tries to perform a database operation, in addition to the first one. The default value of this property is 5 (five). If the value of this property is equal to 0 (zero), only one try is performed by operation. In case there is a database connection loss during any of these tries, that operation is aborted and the remaining attempts are dismissed.

5.7.2.5 ReconnectDelay

The **ReconnectDelay** property determines a delay time (in milliseconds) for an application to try to restore a lost database connection. The default value of this property is 2000 (two seconds).

5.7.2.6 SourceDatabase

A For an **Access**-type database, this is the name of an .mdb file. For a **SQL Server**-type database, this is the name of a SQL Server, concatenated with the chosen database, such as Server/Database. For an **Oracle**-type database, this is the name of a connection. This is a read-only property.

5.7.2.7 SourceType

Indicates the database used by this object. The available options are the following:

- **0 - stAccess**: Access database
- **1 - stOracle**: Oracle database
- **2 - stSqlServer**: SQL Server database

5.7.2.8 TimeOutCommand

9 Contains a delay time for any database operation, before an application generates a *timeout* error. The default value is 180 (three minutes).

5.7.2.9 TimeOutConnection

9 Contains a delay time to perform a database connection, before an application generates a *timeout* error. The default value is 15 seconds.

5.7.2.10 TotalFailedWrites

9 Indicates the total amount of operations on .e3o files that failed and were discarded since the database connection was activated. This property can be automatically zeroed in several situations, such as:

- When deactivating the DBServer object
- If the E3DBEngine process is closed for any reason
- If configuration parameters of the DBServer object's connection were changed

5.7.2.11 UserName

A A login used to connect to a Database. This is a read-only property.

5.7.2.12 UseTransaction

◩ Defines whether a Database Server uses database transactions or not. If this property is set to True, each block of database operations (200 operations), such as Historic, Storage, Formula, and Alarm operations are executed at once, that is, as a single transaction.

5.8 Historic

This section contains information about **methods** and **properties** of the Historic object. This object does not have events associated to it.

5.8.1 Methods

This section contains information about the methods of the Historic object.

5.8.1.1 StartAcquisition

StartAcquisition()

Enables a Historic to record its field values periodically, using a rate specified in the **ScanTime** property. This method can be called at any time after calling the **StopAcquisition** method. The default behavior of this method is to start the application as enabled, that is, this method is always executed internally when starting a Historic. Example:

```
Sub Button1_Click()
  ' When clicking the button, enables the historic.
  Application.GetObject("Hist1").StartAcquisition()
End Sub
```

5.8.1.2 StopAcquisition

StopAcquisition()

Disables acquiring records by period in the Historic, regardless of what value is set in the **ScanTime** property. The recording by period is disabled until the **StartAcquisition** method is called. The default behavior of the Historic object is starting the application with recording disabled. Example:

```
Sub Hist1_OnStartRunning()
  ' Disables Historic as soon as it starts.
  StopAcquisition()
End Sub
```

5.8.1.3 WriteRecord

WriteRecord()

Inserts a new record in the Database. The values are obtained from current values

of each variable specified as Historic field's data sources. This method is used in two situations:

- To write a new data record before the time set for the next recording, when the Historic is enabled by time
- To write a new data set when the Historic is disabled

Example:

```
Sub Tag1_OnValueChange()
  ' Writes a new record in the Historic
  ' when a tag changes its value.
  Application.GetObject("Hist1").WriteRecord()
End Sub
```

5.8.2 Properties

This section contains information about the properties of the Historic object.

5.8.2.1 BackupDiscardInterval

9 Indicates the amount of time units in which backup data is kept on the main table and on the backup table, until it is discarded (for example, to keep data for 24 months on the main table, and more six months on the backup table, the property value must be 30 months). This property works together with the **BackupDiscardTimeUnit** property. The default value of this property is 12 (twelve time units indicated in the **BackupDiscardTimeUnit** property).

NOTE: The total time indicated by combining the **BackupDiscardInterval** an **BackupDiscardTimeUnit** properties must be greater than the time indicated by the **DiscardInterval** and **DiscardTimeUnit** properties.

5.8.2.2 BackupDiscardTimeUnit

The **BackupDiscardTimeUnit** property indicates the time unit in which backup data remains until it is discarded. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

This property works together with the **BackupDiscardInterval** property.

5.8.2.3 CacheSize

9 Defines the record's block size that must be read by the Historic, before sending them to the database. For example, if **CacheSize** is equal to 4 (four), blocks with four records each are sent to the associated Database Server. The valid values for this property must be in a range between 1 and 4. The default value of this property is 1.

NOTE: The record block is sent every second, even when it does not reach the size configured in the **CacheSize** property.

5.8.2.4 CompressedTable

◩ Enables using dead band for data recording. The default value of this property is False.

5.8.2.5 DBServer

A Indicates the Database used by the Historic to create tables and to record data. The default value of this property is an empty **String**.

5.8.2.6 DeadBand

9 This property works together with the **CompressedTable** property. Indicates the calculated value over the last recorded value (a percentage) that defines whether this new value is recorded. If the recorded value is not numeric, changing it makes all values to be recorded.

5.8.2.7 DiscardInterval

9 This property works together with the **DiscardTimeUnit** property. The **DiscardInterval** property indicates the time interval during which Historic data is stored on the database table, until it is discarded. The default value of this property is 1 (one time unit indicated in the **DiscardTimeUnit** property). If this property is set to a value less than the value of the **BackupDiscardInterval** property, E3 automatically sets the value of the **BackupDiscardInterval** property as double the value of the **DiscardInterval** property. This property can be modified at run time.

5.8.2.8 DiscardTimeUnit

This property works together with the **DiscardInterval** property. The **DiscardTimeUnit** property indicates the time unit in which data table is stored, until it is discarded. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days

- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

5.8.2.9 EnableBackupTable

◩ Creates a backup table where discarded data remains for security reasons. If set to True, the table is created. Otherwise, there is no backup table. The default value of this property is False.

5.8.2.10 EnableDiscard

◩ Indicates Historic data discarding after a certain period of time. If set to False, data is stored indefinitely on the table. Otherwise, it is discarded after a certain period of time. The default value of this property is False.

5.8.2.11 EnableQualityLogs

◩ When in True, the Historic starts and E3 generates a record equal to the first collected record, but with bad quality (0) and a time stamp of minus one second.

5.8.2.12 ScanTime

Defines the time interval variation, in milliseconds, that the Historic waits to perform acquisition and recording of a new record on the table. Use this property if there is a need for more or less amount of data generated per second. The default value of this property is 1,000.

5.8.2.13 TableName

A Defines the name of the table used in the Historic.

5.8.2.14 UserTable

◩ When this property is set to True, identifies this Historic as a user type, that is, table data was imported from the database. Otherwise, it is an E3's regular Historic. This is a read-only property.

5.8.2.15 UseTagQuality

◩ If True, the Historic uses the Tag source's quality value. Otherwise, the old evaluation method is used (**0**: uncertain value; **1**: good value).

5.8.2.16 VerificationInterval

9 This property works together with the **VerificationUnit** property to control the time interval that E3 checks for older data, and then discard them. The default value of

this property is 1 (one time unit indicated by **VerificationUnit**).

5.8.2.17 VerificationUnit

This property works together with the **VerificationInterval** property. The **VerificationUnit** property indicates the time unit in which data discard verification is performed. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

5.9 Storage

This section contains information about **methods** and **properties** of the Storage object. This object does not have events associated to it.

5.9.1 Methods

This section contains information about the methods of the Storage object.

5.9.1.1 CreateNewSession

CreateNewSession([*DefaultType*[, *DefaultMinRecTime*[, *DefaultMaxRecTime*[, *DefaultDeadBand*[, *DefaultDeadBandUnit*[, *DefaultScanTime*]]]]]])

Creates a **Session** capable of including data on a Storage, independent of normal acquisition. The optional parameters are used to configure Session Tags, in case they are not informed during creation. They are, respectively:

- **DefaultType**: Data type (**0: Double**, **1: Bit**, **2: String**, or **3: Integer**). If no value is informed, then 0 (**Double**) will be used
- **DefaultMinRecTime**: Minimum time interval between recordings. If no value is informed, then 0 (zero) will be used
- **DefaultMaxRecTime**: Maximum time interval without recordings. If no value is informed, then 3600 will be used
- **DefaultDeadBand**: Dead band. If no value is informed, then 1 (one) will be used
- **DefaultDeadBandUnit**: Tag's dead band unit (**0: Percentage** or **1: Absolute**). If no value is informed, then 1 (**Absolute**) will be used
- **DefaultScanTime**: Scan time. If no value is informed, then 0 (zero) will be used

This methods works even if there are no fields configured in the Storage object.

5.9.1.2 StartAcquisition

StartAcquisition()

Starts or resumes data generation to the Database. The Storage receives notifications about what Tags were modified, and when this happens, it checks whether records were stored or not. When this method is called, the change notification and the recording generation are started or resumed. Example:

```
Sub Button1_Click()
  ' When clicking this button, enables the Storage
  Application.GetObject("Storage1").StartAcquisition()
End Sub
```

5.9.1.3 StopAcquisition

StopAcquisition()

Stops data generation to the Database. The Storage receives notifications about what Tags were modified, and when this happens, it checks whether records were recorded or not. When this method is called, the change notification and the recording generation are stalled. Example:

```
Sub Storage1_OnStartRunning()
  ' Disables the Storage as soon as it starts
  StopAcquisition()
End Sub
```

5.9.2 Properties

This section contains information about the properties of the Storage object.

5.9.2.1 BackupDiscardInterval

9 Indicates the amount of time units during which backup data is kept on the main table and on the backup table, until it is discarded (for example, to keep data for 24 months on the main table, and six more months on the backup table, the value of this property must be 30 months). This property works together with the **BackupDiscardTimeUnit** property. The default value of this property is 12 (twelve time units indicated by **BackupDiscardTimeUnit**).

NOTE: The total time indicated by the combination of **BackupDiscardInterval** and **BackupDiscardTimeUnit** properties must be greater than the time indicated by **DiscardInterval** and **DiscardTimeUnit** properties.

5.9.2.2 BackupDiscardTimeUnit

The **BackupDiscardTimeUnit** property indicates the time unit in which backup data remains stored, until it is discarded. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

This property works together with the **BackupDiscardInterval** property.

5.9.2.3 CacheSize

Defines the block size of records which must be read by the Storage before sending them to the database. For example, if **CacheSize** is equal to 4, blocks with four records each are sent to the associated Database object. The default value of this property is 10.

NOTE: A record block is sent every 1 second, even if it does not reach the size configured in the **CacheSize** property.

5.9.2.4 CompressionRate

Displays the data compression rate obtained up until now.

5.9.2.5 DBServer

Indicates the Database object used by the Storage to create tables and data records. The default value of this property is an empty String.

5.9.2.6 DiscardInterval

This property works together with the **DiscardTimeUnit** property. The **DiscardInterval** property indicates the time interval into which Historic data remains stored on the database table, until it is discarded. The default value of this property is 1 (one time unit indicated by the **DiscardTimeUnit** property). If this property is set to a value less or equal to the value of **BackupDiscardInterval** property, E3 automatically adjusts the value of the **BackupDiscardInterval** property as double the value of the **DiscardInterval** property. This property can be modified at run time.

5.9.2.7 DiscardTimeUnit

This property works together with the **DiscardInterval** property. The **DiscardTimeUnit** property indicates the time unit into which data remains stored until it is discarded. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

5.9.2.8 EnableBackupTable

Creates a backup table where discarded data remains for security reasons. If set to True, the Storage creates this table. Otherwise, there is no backup table. The default value of this property is False.

5.9.2.9 EnableDiscard

Enables data discarding after a certain amount of time. If set to False, data is indefinitely stored on the table. Otherwise, data is discarded after a certain amount of time. The default value of this property is False.

5.9.2.10 Fields

Collection which points to Fields created in the Storage. For each Field, it is possible to view **Name** and **Link** properties, and modify **Type**, **MinRecTime**, **MaxRecTime**, **DeadBand**, **ScanTime**, and **DeadBandUnit** properties.

5.9.2.11 StringFieldSize

This property specifies the maximum size that Storage's **String**-type fields may have (this is the size used when creating a **Value** field of the Strings table).

5.9.2.12 TableName

Defines the name of the table to be used in the Storage.

5.9.2.13 VerificationInterval

This property works together with the **VerificationUnit** property to control the time interval into which E3 checks how old data is, to discard it later. The default value of this property is 1 (one time unit indicated by **VerificationUnit**).

5.9.2.14 VerificationUnit

This property works together with the **VerificationInternal** property. The **VerificationUnit** property indicates the time unit into which data discard checking is performed. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

5.9.3 Storage Fields

This section contains information about **properties** of the Storage Field object. This object does not have events nor methods associated to it.

5.9.3.1 Properties

This section contains information about the properties of the Storage Field object.

5.9.3.1.1 DeadBand

Dead band used to calculate Storage algorithm. Indicates the precision that users are willing to lose on every Field being stored. The higher the value of **DeadBand** property, the more compact the database (less data is recorded). This value can be specified in absolute units, or as a percentage of Tag's current value, according to **DeadBandUnit** property.

5.9.3.1.2 DeadBandUnit

Unit of **DeadBand** property. It can be specified in absolute units, or as a percentage of Tag's current value.

5.9.3.1.3 Link

Determines the data source associated to this Field. This is a read-only property.

5.9.3.1.4 MaxRecTime

Maximum difference, in seconds, between timestamps of two consecutive records stored on a database, that is, the maximum time with any data being recorded. For example, if a Tag value is not varying, yet its current value must be written to the database when the number of seconds configured in **MaxRecTime** is reached. This behavior can be disabled by setting property value to 0 (zero).

5.9.3.1.5 MinRecTime

9 Minimum difference, in milliseconds, between timestamps of any two records with the same quality stored on a database, that is, the minimum time for a new data to be recorded. This parameter is used to limit the number of records written to a database, in case of a Tag undergoing sudden variations on its value. This behavior can be disabled by setting property value to 0 (zero).

5.9.3.1.6 Name

A Property returning the name configured for this Field in the Storage object. By using this name, it is possible to search for items in the collection.

5.9.3.1.7 ScanTime

9 Returns or sets Field's scan time (in milliseconds), that is, how often a Tag value is sent to the compression algorithm, in case it is not varying. If this property's value is equal to 0 (zero), the value of **MaxRecTime** property is used for the same purpose.

5.9.3.1.8 Type

Returns the object type in the Storage format (**0: Double**, **1: Bit**, **2: String**, **3: Integer**). This is a read-write property, but it only accepts changes while data collecting for this Field has not started yet.

NOTE: **Bit**, **String**, and **Integer** datatypes are not submitted to Storage's compression algorithm. Therefore, when there is a variation on Field's quality or value, the value is recorded on the database. **DeadBand**, **DeadBandUnit**, **MaxRecTime**, and **MinRecTime** properties do not affect these datatypes, because they are exclusive to the compression algorithm.

5.9.4 Storage Session

This section contains information about **methods** of the Storage Session object. This object does not have events nor properties associated to it.

5.9.4.1 Methods

This section contains information about the methods of the Storage Session object.

5.9.4.1.1 AddField

AddField(*FieldName*[, *Type*, *MinRecTime*, *MaxRecTime*, *DeadBand*, *DeadBandUnit*, *ScanTime*])

This method is responsible for inserting temporary Tags on the Session's structure.

If its optional parameters are not informed, the values defined during the creation of the Session using the Storage's **CreateNewSession** method will be used. This method's parameters are described on the following table.

Parameters of the AddField method

PARAMETER	DESCRIPTION
FieldName	Name of the temporary Tag (mandatory).
Type	Tag type (optional). Possible values are **0: Double**, **1: Bit**, **2: String**, or **3: Integer**. The default value of this parameter is 0 (**Double**).
MinRecTime	Minimum time interval between recordings (optional). The default value of this parameter is 0 (zero).
MaxRecTime	Maximum time interval without recordings (optional). The default value of this parameter is 3600.
DeadBand	Temporary Tag's dead band (optional). The default value of this parameter is 1 (one).
DeadBandUnit	Temporary Tag's dead band unit (optional). Possible values are **0: Percentage** or **1: Absolute**. The default value of this parameter is 1 (**Absolute**).
ScanTime	Temporary Tag's scan time (optional). The default value of this parameter is 0 (zero).

This method returns True if the Tag was correctly added to the Session, and False otherwise.

5.9.4.1.2 AddValue

AddValue(*FieldName*, *Timestamp*, *Quality*, *Value*)

Adds a value to a temporary Tag on the Session. This method's parameters are described on the following table.

Parameters of the AddValue method

PARAMETER	DESCRIPTION
FieldName	Name of the field to which the value will be added. This name must exist on the original Storage's configuration, or else it must have been previously added using the **AddField** method.
Timestamp	Time stamp of the value to be added.
Quality	Quality of the value to be added.

PARAMETER	DESCRIPTION
Value	Value to be added.

This method returns True if the value was correctly added, and False otherwise.

5.9.4.1.3 Commit

Commit()

Stores all Session data kept in memory on the Storage's database. This method returns True if data was correctly stored, and False otherwise.

5.10 Formulas

This section contains information about **methods** and **properties** of the Formula object. This object does not have events associated to it.

5.10.1 Methods

This section contains information about the methods of the Formula object.

5.10.1.1 CreateUnit

CreateUnit(*UnitName*)

Creates a unit on the Formula table. This method has the *UnitName* parameter which determines the name of the unit to be created. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, creates a new Unit
  Application.GetObject("Formula1").CreateUnit("Unit2")
End Sub
```

5.10.1.2 CreateValue

CreateValue(*ValueName*)

Creates a value set on the Formula table. This method has the *ValueName* parameter, which determines the name of the set to be created. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, creates a new Value
  Application.GetObject("Formula1").CreateValue("Template5")
End Sub
```

5.10.1.3 DeleteUnit

DeleteUnit(*UnitName*)

Deletes a unit on the Formula table. This method has the *UnitName* parameter, which informs the name of the unit to be deleted. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, deletes the unit
  Application.GetObject("Formula1").DeleteUnit("Unit2")
End Sub
```

5.10.1.4 DeleteValue

DeleteValue(*ValueName*)

Deletes a value set on the Formula table. This method has the *ValueName* parameter, which informs the value set to be deleted. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, deletes a value set
  Application.GetObject("Formula1").DeleteValue("Template5")
End Sub
```

5.10.1.5 FindUnit

FindUnit(*UnitName*)

Checks if a certain unit exists on the Formula database. This method has the *UnitName* parameter, which determines the name of the unit to be found. This method returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, displays a message box
  ' with the result
  MsgBox(Application.GetObject("Formula1")._
    FindUnit("Unit2"))
End Sub
```

5.10.1.6 FindValue

FindValue(*ValueName*)

Checks if a certain value set exists on Formula database. This method has the *ValueName* parameter, which informs the name of the set to be checked. Returns True if the operation is successful. Otherwise returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button displays a message box
  ' with the result
  MsgBox CStr(Application.GetObject("Formula1")._
    FindValue("Template5"))
End Sub
```

5.10.1.7 GetUnitData

GetUnitData(*UnitName*, *TemplateName*, *Val*)

Places on the variable indicated by *Val* the Tag associated to the *UnitName* unit of the *TemplateName* template. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim whatTag, whatFormula
  Application.GetObject("Formula1").GetUnitData _
    "Unit1", "Template2", var1
End Sub
```

5.10.1.8 GetValueData

GetValueData(*ValueName*, *TemplateName*, *Val*)

Places on the *Val* variable the value of the *ValueName* value set which is associated to the *TemplateName* template. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim Value, whatFormula
  Application.GetObject("Formula1").GetValueData _
    "Value4", "Template2", var1
End Sub
```

5.10.1.9 LoadFormulaValuesQuiet

LoadFormulaValuesQuiet(*UnitName*, *ValueName*)

Loads a value set to a destination unit, without displaying a message. This method has the *UnitName* parameter, which determines the name of the unit and the *ValueName* parameter, which determines the name of the value set. Returns True if the operation is successful, and False on failure (which does not necessarily mean a script error). Example:

```
Sub Button1_Click()
  Application.GetObject("Formula1").LoadFormulaValuesQuiet _
    "Unit3", "Value1"
End Sub
```

NOTE: This method is also acessible through the **Viewer** object.

5.10.1.10 RenameUnit

RenameUnit(*UnitName*, *NewUnitName*)

Renames a certain existing unit on the Formula table. Returns True if the operation is successful, or False otherwise. This method has the *UnitName* parameter, which determines the name of the unit to be found, and the *NewUnitName* parameter, which informs the new unit name. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, renames the unit
  Application._
    GetObject("Formula1").RenameUnit "Unit2", "Unit3"
End Sub
```

5.10.1.11 RenameValue

RenameValue(*ValueName*, *NewValueName*)

Renames a certain set of values which exists in the Formula table. Returns True if the operation is successful or False, otherwise. This method has the *ValueName* parameter, which is the name of the set of values and the *NewValueName* parameter, which is the new name of the set of values. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1").RenameValue "Template5",
"TemplateABC"
End Sub
```

5.10.1.12 SaveFormulaValues

SaveFormulaValues(*UnitName*, *ValueName*[, *IgnoreErrors*])

This method saves the current values of the Tags from an origin unit into a set of values in the Formula table. This method does not check limits, in case the template has an absolute type restriction. The *UnitName* parameter is the name of the origin unit and the *ValueName* parameter is the set of values which is saved. Returns True if the operation is successful. Otherwise, returns False. The *IgnoreErrors* parameter, when in True, forces all values to be saved, even if there are errors in the Formula links. However, its default value is False. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1") SaveFormulaValues "Unit1",
"Value1"
End Sub
```

5.10.1.13 SetUnitData

SetUnitData(*UnitName*, *TemplateName*, *Data*)

Loads the identified Tag to a certain template, on a certain unit, in the Formula

table. Returns True if the operation is successful or False, otherwise. This method has the *UnitName* parameter, which is the name of the unit, the *TemplateName* parameter, which is the name of the Tag template, and the *Data* parameter, which is the name of the variable that contains the Tag name. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1").SetUnitData "Unit2",
"Template5", 50
End Sub
```

5.10.1.14 SetValueData

SetValueData(*ValueName*, *TemplateName*, *Data*)

Changes the value referring to a template defined for a certain set of values. This method checks for limit values, returning True if the operation is successful or False, otherwise. This method has the *ValueName* parameter, which is the name of the set of values, the *TemplateName* parameter, which determines the template name, and the *Data* parameter, which is the name of the variable that holds the value. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1").SetValueData "Unit2",
"Template1", 100
End Sub
```

NOTE: For more information about the **SetValueData** method, please check the articles **KB 112**, **KB 1246**, and **KB 4946** from the **Elipse Knowledgebase** website.

5.10.2 Properties

This section contains information about the properties of the Formula object.

5.10.2.1 DBServer

A Indicates the name of the database where Formula information is stored, that is, units and value sets. The default value of this property is an empty **String**.

5.10.2.2 ImmediateExecute

◩ When enabled, the Formula object writes its records directly to the Database, without passing through operation queues (.e3i and .e3o files). This makes database operations to be viewed faster.

5.10.2.3 TableName

A Indicates the name of the tables where Formula information is stored. The default value of this property is an empty **String**.

5.11 Alarms

This section contains information about events, methods, and properties of the Alarm Configuration, Alarm Area, Alarm Source, and Alarm Server objects.

5.11.1 Alarm Configuration

The Alarm Configuration object does not contain specific events, methods, or properties, only general ones. These can be viewed on section **General Events, Methods, and Properties of Objects**.

5.11.2 Alarm Area

This section contains information about **properties** of the Alarm Area object. This object does not have events nor methods associated to it.

5.11.2.1 Properties

This section contains information about the properties of the Alarm Area object.

5.11.2.1.1 ActiveAlarms

Determines the number of active alarms in the Area. This is a read-only property and its default value is 0 (zero).

5.11.2.1.2 ActiveHighAlarms

Indicates the number of active alarms with high severity. This is a read-only property.

5.11.2.1.3 ActiveHighNACKAlarms

Indicates the number of non-acknowledged alarms with high severity. This is a read-only property.

5.11.2.1.4 ActiveLowAlarms

Indicates the number of active alarms with low severity. This is a read-only property.

5.11.2.1.5 ActiveLowNACKAlarms

Indicates the number of non-acknowledged alarms with low severity. This is a read-only property.

5.11.2.1.6 ActiveMedAlarms

Indicates the number of active alarms with medium severity. This is a read-only property.

5.11.2.1.7 ActiveMedNACKAlarms

Indicates the number of non-acknowledged alarms with medium severity. This is a read-only property.

5.11.2.1.8 ActiveNACKAlarms

Indicates the number of non-acknowledged alarms inside an object. This is a read-only property.

5.11.2.1.9 Alarm

Indicates that there are active alarms inside the Area. If this option is set to True, there is at least one active alarm inside the Area, and the property **ActiveAlarms** performs a reading from server, then indicating the amount of active alarms. Otherwise, property **ActiveNACKAlarms** performs a reading of non-acknowledged alarms. This is a read-only property.

5.11.2.1.10 AlarmVerify

Enables a check on all alarms inside the object. After enabling that check (True), if the value of property **ActiveAlarms** is greater than 0 (zero), server then checks active alarms, as well as non-acknowledged ones, listing these last ones in the property **ActiveNACKAlarms**. This property is useful to avoid a cascade effect on some applications, where the occurrence of an event triggers a large amount of related alarms.

5.11.2.1.11 InactiveHighNACKAlarms

Indicates the number of active and unacknowledged alarms with high severity. This is a read-only property.

5.11.2.1.12 InactiveLowNACKAlarms

Indicates the number of active and unacknowledged alarms with low severity. This is a read-only property.

5.11.2.1.13 InactiveMedNACKAlarms

Indicates the number of active and unacknowledged alarms with medium severity. This is a read-only property.

5.11.2.1.14 InactiveNACKAlarms

9 Determines the total amount of active and unacknowledged alarms. This is a read-only property.

5.11.2.1.15 UserFields

Returns an object that is a collection of Alarm's User Fields of an Alarm Area. Please check the item **Collection of Alarm's User Fields** for more information about the collection of objects returned by this property.

5.11.3 Alarm Source

This section contains information about common **methods** and **properties** of the Alarm Source object. This object does not have common events associated to it.

NOTE: When an Alarm Source object is disabled or deactivated, the **ActiveNACKAlarm**, **Alarm**, **CurrentSeverity**, **CurrentSubConditionName**, **FormattedValue**, **RawAlarm**, and **Value** properties assume their default values, indicating that the Alarm Source is not linked to an active alarm message. In this case, the values of the **Alarm** and **ActiveNACKAlarm** are propagated to the counters of upper areas.
For these properties to receive a value other than the default, it is necessary that the following conditions are present:

- The Alarm Source must be enabled and active
- The **Alarm Area** (and all its hierarchically superior objects) must be enabled and active
- The **Alarm Server** must be active

5.11.3.1 Common Methods

This section contains information about the common methods of the Alarm Source objects.

5.11.3.1.1 Ack

Ack([*ActorID*])

Acknowledges an alarm configured in the Alarm Source object. This method returns a **Boolean** indicating whether the operation has been successful or not. The *ActorID* parameter informs the name of the user responsible for alarm acknowledgement. This is an optional parameter and, if omitted, assumes the user logon of the Viewer, "Anonymous" if there is no user logged on, or "System" if the method call started at the server.

5.11.3.1.2 GetAlarm

GetAlarm()

Returns an object that grants access to each type of alarm specific settings. This allows checking or modifying any alarm properties at run time. Depending on the type of alarm, this method returns the following properties:

- **Analog Alarm**: Responsible for digital alarm settings

Digital Alarm Properties

ITEM	DESCRIPTION
DigitalReturnMessageText	Digital alarm returning message.
Digital	Enables or disables digital alarm checking.
DigitalLimit	Digital alarm limit.
DigitalMessageText	Digital alarm message text.
DigitalSeverity	Digital alarm severity. Set of values: • **0**: High • **1**: Medium • **2**: Low
DigitalAckRequired	Acknowledgement required for this type of alarm (digital).

- **Analog Alarm:** Responsible for the definition of the alarm level. Properties of this object (it has four alarm levels):

Analog Alarm properties

ITEM	DESCRIPTION
LevelDeadBand	Dead band for alarm level.
LevelReturnMessageText	Returns alarm's message.

LoLo Alarm properties

ITEM	DESCRIPTION
LoLo	Enables or disables LoLo alarm checking.
LoLoLimit	LoLo alarm limit.
LoLoMessageText	LoLo alarm message text.
LoLoSeverity	LoLo alarm severity. Set of values: • **0**: High • **1**: Medium • **2**: Low

ITEM	DESCRIPTION
LoLoAckRequired	Acknowledgement required for this alarm level (LoLo).

Lo Alarm properties

ITEM	DESCRIPTION
Lo	Enables or disables the Lo alarm checking.
LoLimit	Lo alarm limit.
LoMessageText	Lo alarm message text.
LoSeverity	Lo alarm severity. Set of values: • **0**: High • **1**: Medium • **2**: Low
LoAckRequired	Acknowledgement required for this alarm level (Lo).

Hi Alarm properties

ITEM	DESCRIPTION
Hi	Enables or disables the High alarm checking.
HiLimit	Hi alarm limit.
HiMessageText	Hi alarm message text.
HiSeverity	Hi alarm severity. Set of values: • **0**: High • **1**: Medium • **2**: Low
HiAckRequired	Acknowledgement required for this alarm level (Hi).

HiHi Alarm properties

ITEM	DESCRIPTION
HiHi	Enables or disables the HiHi alarm checking.
HiHiLimit	HiHi alarm limit.
HiHiMessageText	HiHi alarm message text.
HiHiSeverity	HiHi alarm severity. Set of values: • **0**: High • **1**: Medium • **2**: Low

ITEM	DESCRIPTION
HiHiAckRequired	Acknowledgement required for this alarm level (HiHi).

- **Rate of Change Alarm**: Responsible for the settings of a **Rate Of Change**-type alarm

Rate of Change Alarm properties

ITEM	DESCRIPTION
ROCReturnMessageText	Rate of Change return message.
ROC	Enables or disables Rate of Change checking.
ROCLimit	Rate of Change alarm limit. For this alarm to occur, it is enough that the value of the associated Tag exceeds this value in one second.
ROCMessageText	Rate of Change alarm message text.
ROCSeverity	Rate of Change alarm severity. Set of values: • **0**: High • **1**: Medium • **2**: Low
ROCAckRequired	Acknowledgement required for this type of alarm (rate of change).

- **Dead Band Alarm**: Responsible for dead band alarm settings

Dead Band Alarm Properties

ITEM	DESCRIPTION
DeadBandSetPoint	Dead Band alarm limit. Every time the value of the associated Tag exceeds the value of this property up or down the **DeadBandLimit** value, this alarm occurs.
DeadBandReturnMessageText	Dead band alarm return message.
DeadBand	Enables or disables dead band alarm checking.
DeadBandLimit	Dead band alarm limit.
DeadBandMessageText	Dead band alarm message text.
DeadBandSeverity	Severity of dead band alarm. Set of values: • **0**: High • **1**: Medium • **2**: Low
DeadBandAckRequired	Acknowledgement required for this type of alarm (dead band).

Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, AlarmSource BatteryLevel
  ' Lo alarm level is changed
  Application.GetObject("ConfigAlarms1.Area1.BatteryLevel")_
    .GetAlarm().LoLimit = 10.2
End Sub
```

NOTE: specific properties for each alarm source type can be accessed directly via scripts or links, as well as visualized in the object's Property List, making their editing via **GetAlarm** method no longer mandatory.

5.11.3.2 Common Properties

This section contains information about the common properties of the Alarm Source objects.

5.11.3.2.1 ActiveNACKAlarm

◩ If set to True, indicates that the Source was not acknowledged since last activation. This is a read-only property. The default value of this property is False.

5.11.3.2.2 Alarm

◩ If set to True, indicates the Alarm's active condition. The default value of this property is False.

5.11.3.2.3 AlarmVerify

◩ If set to True, enables Alarm source's check (that is, generating the Alarm).

5.11.3.2.4 AreaNameOverride

A Determines an alternative name for the Area that contains the Alarm Source. The default value of this property is an empty **String**.

NOTES:

- When this property is an empty **String**, the name of the Alarm Source is composed by the names of the Area objects hierarchically above it
- This property, even when filled in, does not influence the Alarm Area counters hierarchically above it, which keep counting the alarms of this Alarm Source
- If this property is changed at run time, the new value is only effective when the next alarm on this Alarm Source occurs

5.11.3.2.5 CurrentSeverity

Indicates the last severity of the active alarm, that is:

- **0**: High
- **1**: Medium
- **2**: Low

The default value of this property is -1, indicating that the Alarm Source is not active.

5.11.3.2.6 CurrentSubConditionName

Determines the name of the last active alarm condition. The available options for this property are:

Available options for CurrentSubConditionName

OPTION	DESCRIPTION
LOLO	Analog Alarm on LOLO range.
LO	Analog Alarm on LO range.
HI	Analog Alarm on HI range.
HIHI	Analog Alarm on HIHI range.
DB	Dead Band Alarm.
ROC	Rate Of Change Alarm.
DIG	Digital Alarm.

The default value of this property is an empty **String**.

5.11.3.2.7 Delay

Specifies a delay time for the Alarm (in ms) when entering, as well as leaving, the condition. When this property is set to 0 (which will always be the default value), no delay is applied. When different from 0, the alarm is only activated or deactivated if it remains on the same condition for a time greater than or equal to the one specified.

5.11.3.2.8 DoubleAckRequired

When configured as True, indicates that the alarm can be acknowledged when active, as well as when it is inactive, that is, can be doubly acknowledged. When configured as False, indicates that the alarm can only be acknowledged once and, when acknowledged, it leaves the alarm list. Alarms that do not need acknowledgment (by using the **AckRequired** property) does not allow that type of

customization. Applications prior to version 2.5 have this property set to False.

5.11.3.2.9 Event

◩ When configured as True, indicates that this is an **Event**-type alarm. An **Event**-type alarm, when activated, is acknowledged by the "System" user. Therefore, when it is acknowledged nothing happens (it has no effect), as well as it does not increment the number of active alarms, nor the number of unacknowledged alarms. It cannot be modified at run time.

5.11.3.2.10 Format

A The **Format** property specifies what type of format is applied to the object. It allows changing the way data is displayed, without changing data behind it. This property can be manually edited or configured using the Format window. Its usage is similar to formats used on spreadsheets, following the same basic syntax. The following data types are supported:

Data types supported by Format

DATA	DESCRIPTION
Numerical	Decimal, scientific, hexadecimal, binary, and octal output.
Text	General text.
Boolean	**Boolean** values.
Date/Time	Gregorian calendar.

5.11.3.2.11 FormattedValue

A Contains the value of the alarm in the **Value** property, according to the **Format** property. This is a read-only property. The default value of this property is Null.

5.11.3.2.12 RawAlarm

◩ Indicates whether the alarm should be active, independent of the waiting time. When the waiting time is equal to 0 (zero), the value of **RawAlarm** is the same as the **Alarm** property. This is a read-only property. The default value of this property is False.

5.11.3.2.13 Source

A Contains the expression that must be evaluated to determine whether the alarm should occur.

5.11.3.2.14 UserFields

Returns an object that is a collection of Alarm's User Fields of an Alarm Source. Please check the item **Collection of Alarm's User Fields** for more information about the collection of objects returned by this property.

5.11.3.2.15 Value

Contains the value that was evaluated to determine whether the alarm occurs. The default value of this property is Null.

5.11.3.3 Analog Alarm Source

This section contains information about **properties** of the Analog Alarm Source object. This object does not have events nor methods associated to it.

When the value of the **Event** property (common to all Alarm Sources) is set to True, the **HiEvent**, **HiHiEvent**, **LoEvent**, and **LoLoEvent** properties cannot be modified (all alarm subconditions behave as events).

5.11.3.3.1 Properties

This section contains information about the properties of the Analog Alarm Source object.

5.11.3.3.1.1 Hi

Enables or disables a check on **Hi**-type alarms.

5.11.3.3.1.2 HiAckRequired

Indicates that a **Hi**-type alarm requires acknowledgment.

5.11.3.3.1.3 HiEvent

Defines whether the Alarm's **Hi** subcondition should be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in True. In addition, this property cannot be modified at run time.

5.11.3.3.1.4 HiHi

Enables or disables a check on **HiHi**-type alarms.

5.11.3.3.1.5 HiHiAckRequired

◩ Indicates that a **HiHi**-type alarm requires acknowledgment.

5.11.3.3.1.6 HiHiEvent

◩ Defines whether the Alarm's **HiHi** subcondition should be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains as True. In addition, this property cannot be modified at run time.

5.11.3.3.1.7 HiHiLimit

9 Sets the limit of a **HiHi**-type alarm.

5.11.3.3.1.8 HiHiMessageText

A Sets the text message of a **HiHi**-type alarm limit.

5.11.3.3.1.9 HiHiSeverity

Indicates the level of importance of a **HiHi**-type alarm. The options are:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.3.1.10 HiLimit

9 Indicates the level in which a **Hi**-type alarm is activated.

5.11.3.3.1.11 HiMessageText

A Sets the text message of a **Hi**-type alarm limit.

5.11.3.3.1.12 HiSeverity

Indicates the level of importance of a **Hi**-type alarm. The options are:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.3.1.13 LevelDeadBand

9 Dead band for the alarm level limits.

5.11.3.3.1.14 LevelReturnMessageText

A Sets the return message of the alarm level.

5.11.3.3.1.15 Lo

◩ Enables or disables a check on **Lo**-type alarms.

5.11.3.3.1.16 LoAckRequired

◩ Indicates that a **Lo**-type alarm requires acknowledgment.

5.11.3.3.1.17 LoEvent

◩ Defines whether the Alarm's **Lo** subcondition should be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in True. In addition, this property cannot be modified at run time.

5.11.3.3.1.18 LoLimit

9 Indicates the level in which a **Lo**-type alarm is activated.

5.11.3.3.1.19 LoLo

◩ Enables or disables a check on **LoLo**-type alarms.

5.11.3.3.1.20 LoLoAckRequired

◩ Indicates that a **LoLo**-type alarm requires acknowledgment.

5.11.3.3.1.21 LoLoEvent

◩ Defines whether the Alarm's **LoLo** subcondition should be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in True. In addition, this property cannot be modified at run time.

5.11.3.3.1.22 LoLoLimit

9 Indicates the level in which a **LoLo**-type alarm is activated.

5.11.3.3.1.23 LoLoMessageText

A Sets the text message of a **LoLo**-type alarm limit.

5.11.3.3.1.24 LoLoSeverity

Indicates the level of importance of a **LoLo**-type alarm. The options are:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.3.1.25 LoMessageText

A Sets the text message of a **Lo**-type alarm limit.

5.11.3.3.1.26 LoSeverity

Indicates the level of importance of a **Lo**-type alarm. The options are:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.4 Digital Alarm Source

This section contains information about **properties** of the Digital Alarm Source object. This object does not have events nor methods associated to it.

5.11.3.4.1 Properties

This section contains information about the properties of the Digital Alarm Source object.

5.11.3.4.1.1 Digital

◩ Enables or disables a check on Digital Alarms.

5.11.3.4.1.2 DigitalAckRequired

◩ Indicates the need of acknowledgment for the Digital Alarm.

5.11.3.4.1.3 DigitalLimit

9 Limit for the Digital Alarm.

5.11.3.4.1.4 DigitalMessageText

A Message text of the Digital Alarm.

5.11.3.4.1.5 DigitalReturnMessageText

A Return message of the Digital Alarm.

5.11.3.4.1.6 DigitalSeverity

Severity of the Digital Alarm. Set of values:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.5 Dead Band Alarm Source

This section contains information about **properties** of the Dead Band Alarm Source object. This object does not have events nor methods associated to it.

5.11.3.5.1 Properties

This section contains information about the properties of the Dead Band Alarm Source object.

5.11.3.5.1.1 DeadBand

◩ Enables or disables a check on Dead Band Alarms.

5.11.3.5.1.2 DeadBandAckRequired

◩ Indicates the need of acknowledgment for the Dead Band Alarm.

5.11.3.5.1.3 DeadBandLimit

9 Limit for the Dead Band Alarm.

5.11.3.5.1.4 DeadBandMessageText

A Text of the Dead Band Alarm message.

5.11.3.5.1.5 DeadBandReturnMessageText

A Return message of the Dead Band Alarm.

5.11.3.5.1.6 DeadBandSetpoint

A Alarm's dead band limit. Each time the associated Tag value surpasses this property value more or less the **DeadBandLimit** value, the alarm occurs.

5.11.3.5.1.7 DeadBandSeverity

9 Importance of the Dead Band Alarm. Set of values are:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.6 Rate Of Change Alarm Source

This section contains information about **properties** of the Rate Of Change Alarm Source object. This object does not have events nor methods associated to it.

5.11.3.6.1 Properties

This section contains information about the properties of the Rate Of Change Alarm Source object.

5.11.3.6.1.1 ROC

◩ Enables or disables a check on Rate Of Change Alarms.

5.11.3.6.1.2 ROCAckRequired

◩ Indicates a need of acknowledgment for the Rate Of Change Alarm.

5.11.3.6.1.3 ROCLimit

9 Limit for the Rate Of Change Alarm. For this Alarm to occur, it is enough that the associated Tag value surpasses this value by one second.

5.11.3.6.1.4 ROCMessageText

A Message text of the Rate Of Change Alarm.

5.11.3.6.1.5 ROCReturnMessageText

A Return message of the Rate Of Change Alarm.

5.11.3.6.1.6 ROCSeverity

Importance of the Rate Of Change Alarm. Set of values are:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.7 Discrete Alarm Source

This section contains information about **properties** of the Discrete Alarm Source object. This object does not have events nor methods associated to it.

When the value of the **Event** property (common to all Alarm Sources) is set to True, the **Kind** property of objects belonging to the collection of Subconditions of a Discrete Alarm Source cannot be modified (all alarm subconditions behave as events).

5.11.3.7.1 Properties

This section contains information about the properties of the Discrete Alarm Source object.

5.11.3.7.1.1 DiscreteReturnMessageText

A Returns or modifies the return message of the Discrete Alarm Source. If any object in the collection of **Subconditions** has its **Kind** property set to **2** (Return), the **Message** property of that object is then used, instead of the value defined in **DiscreteReturnMessageText**.

5.11.3.7.1.2 SubConditions

Returns an object that is a collection of Subconditions of the Discrete Alarm Source. Check the item **Collection of Subconditions** for more information about the collection of objects returned by this property.

5.11.3.7.2 Collection of Subconditions

This section contains information about **methods** and **properties** common to the collection of Subconditions returned by the **SubConditions** property of a Discrete Alarm Source.

5.11.3.7.2.1 Common Methods

This section contains information about the methods common to the collection of Subconditions of a Discrete Alarm Source.

AddSubCondition

AddSubCondition([*Name*, *Caption*, *Message*, *Kind*, *AckRequired*, *Severity*, *Value*])

Adds a Subcondition object to the collection of Subconditions. This method has the following optional parameters:

Parameters of AddSubCondition method

PARAMETER	DESCRIPTION
Name	The object's name. Corresponds to the Subcondition object's **Name** property. If it is omitted, the Subcondition is then created with the name "Subcondition". If the value passed on this parameter already exists in the collection, it is automatically incremented.
Caption	The object's description. Corresponds to the Subcondition object's **Caption** property.
Message	The Subcondition's message text. Corresponds to the Subcondition object's **Message** property.
Kind	The type of behavior of this Subcondition. Possible values for this parameter are: **0** - Alarm (default); **1** - Event; **2** - Return. Corresponds to the Subcondition object's **Kind** property.
AckRequired	Indicates whether this Subcondition requires acknowledgement. Corresponds to the Subcondition object's **AckRequired** property. The default value of this property is True.
Severity	The type of severity of this Subcondition. Possible values for this parameter are: **0** - High; **1** - Medium (default); **2** - Low. Corresponds to the Subcondition object's **Severity** property.

PARAMETER	DESCRIPTION
Value	Parameter containing a value that will be evaluated to determine whether the alarm occurs. Corresponds to the **Value** property, common to all Alarm Sources.

Item

Item(*Index*)

Returns a reference to a Subcondition object, indicated by *Index*. This parameter can be an index in the collection (starting at 1), or the object's name (the **Name** property).

RemoveSubCondition

RemoveSubCondition(*Index*)

Removes the Subcondition object indicated by *Index*. This parameter can be an index in the collection (starting at 1), or the object's name (the **Name** property).

5.11.3.7.2.2 Common Properties

This section contains information about the properties of the collection of Subconditions of a Discrete Alarm Source.

Count

9 Returns the number of children objects (items) of a collection of Subconditions. This property works together with the **Item** method. If the collection has no children objects, returned value is 0 (zero).

5.11.3.7.2.3 SubCondition

This section contains information about **properties** of **Subcondition**-type objects inside the collection returned by the **SubConditions** property of a Discrete Alarm Source. This object does not have events nor methods associated to it.

Properties

This section contains information about the properties of the Subcondition object.

AckRequired

◩ Indicates whether this Subcondition object requires acknowledgement.

Caption

A Subcondition's description.

Enabled

◩ Enables or disables the Subcondition.

Kind

Indicates Subcondition's behavior. Possible values for this property are the following:

- **0**: Alarm
- **1**: Event
- **2**: Return

If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in **1** (Event). In addition, this property cannot be modified at run time.

Limit

Defines Alarm Source's value to generate the Subcondition.

Message

The event message when the Subcondition is active. If the **Kind** property is set to **2** (Return), this property is considered as the alarm's return message.

Name

Subcondition object's name. This value is case-insensitive.

Severity

The type of severity of this Subcondition. Possible values for this property are:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.4 Alarm Server

This section contains information about **methods** and **properties** of the Alarm Server object. This object does not events associated to it.

5.11.4.1 Methods

This section contains information about the methods of the Alarm Server object.

5.11.4.1.1 AckAllAlarms

AckAllAlarms([*ActorID*])

Acknowledges all alarms in the server, regardless of their area. This method returns

a **Boolean** indicating whether the operation has been successful or not. The *ActorID* parameter informs the name of the user responsible for acknowledging the alarm. This is an optional parameter and, if omitted, assumes the user logged on the Viewer, "Anonymous" if there is no user logged on, or "System" if the method's call started at the server. Example:

```
Sub Button1_Click()
  ' When clicking the button, it acknowledges all alarms
  Application.GetObject("AlarmServer1")._
    AckAllAlarms (Application.User)
End Sub
```

5.11.4.1.2 AckArea

AckArea(*Area*[, *ActorID*])

Acknowledges the alarms in a given area. This method returns a **Boolean** indicating whether the operation has been successful or not. The *Area* parameter specifies the name of the area(s) which alarms are acknowledged, by matching the beginning of their names. For example, AckArea("ANA") acknowledges alarms in areas named "ANALOG", "ANA.AREA2", etc. If this parameter is empty, the method behaves like **AckAllAlarms**. The *ActorID* parameter informs the name of the user responsible for acknowledging the alarm. This is an optional parameter and, if omitted, assumes the user logged on the Viewer, "Anonymous" if there is no user logged on, or "System" if the method's call started at the server. Example:

```
Sub Button1_Click()
  'When clicking the button, it acknowledges Area1 alarms
  Application.GetObject("AlarmServer1")._
    AckArea "Area1", Application.User
End Sub
```

5.11.4.1.3 LogTrackingEvent

LogTrackingEvent(*Message*[, *ActorID*], *Area*, *Severity*, *EventTime*, *Source*, *EventCategory*, *EventType*, *UserFields*, *AlarmSourceName*, *FullAlarmSourceName*)

Simulates an event or an alarm and sends it straight to Alarm Server's database, without passing througn an E3Alarm. This is why this event cannot be seen on an E3Alarm, nor can the alarm be acknowledged.

Each parameter allows specifying the value of the field with the same name in the event. Event fields are fulfilled according to method parameters.

Available options for the LogTrackingEvent parameters

NAME	DESCRIPTION
Message	Parameter with the content of the text configured on the Alarm Source. If omitted, it assumes an empty **String**.

NAME	DESCRIPTION
ActorID	Optional text parameter specifying the operator that accepted the alarm. If omitted, it assumes the user logged on the Viewer, "Anonymous" if there is no user logged on, or "System" if the method's call started at the server.
Area	Text parameter with the name of the Area where the Alarm Source belongs to. If omitted, it assumes an empty value.
Severity	Numeric parameter informing event severity. If omitted, it assumes 0 (high severity).
EventTime	Date and time parameter informing when the event occurred. If omitted, it assumes the time stamp of the moment of method's call.
Source	Text parameter that specifies the link used to evaluate alarm conditions. If omitted, it assumes an empty **String**.
EventCategory	Text parameter with values for categories of events. If omitted, it assumes an empty **String**.
EventType	Text parameter that specifies the type of the event. For alarms, it assumes the value "Condition". If omitted, it assumes "Tracking".
UserFields	**Array**-type parameter, where each position assumes the value of the user-specified field.
AlarmSourceName	Text parameter with the name of the Alarm Source. If omitted, it assumes an empty **String**.
FullAlarmSourceName	Text parameter with the Alarm Source's full path, including Area, Alarm Configuration, and possible Folders where the Alarm is inserted. If omitted, it assumes an empty **String**.

The other event fields cannot be specified and always assume the following values:

- **CurrentValue**: 0.0
- **Quality**: ""
- **ConditionActive**: 0 (False)
- **ConditionName**: ""
- **SubConditionName**: ""

- **Acked**: 1 (True)
- **AckRequired**: 0 (False)
- **Enabled**: 1 (True)
- **EventTimeUTC**: *Always equal to **EventTime** (just like in Alarm Events)
- **ChangeMask**: 0
- **Cookie**: 0

NOTE: This method fails if the Alarm Server's **Store alarm event in the database** option is set to False, or when a database error occurs.

Example:

```
Sub CommandButton1_Click()
  ' In parameter UserFields, the element value will be shown
  ' for each element
  Application.GetObject("AlarmServer1").LogTrackingEvent_
    "Button clicking", Application.User, "Operation", 2, ,_
    "Button1", , , array(1, 2, "a", "b")
End Sub
```

5.11.4.2 Properties

This section contains information about the properties of the Alarm Server object.

5.11.4.2.1 ActiveAlarms

Determines the total amount of active alarms in the application. This is a read-only property.

5.11.4.2.2 ActiveHighAlarms

Indicates the number of active high severity alarms. This is a read-only property.

5.11.4.2.3 ActiveHighNACKAlarms

Indicates the number of unacknowledged high severity alarms. This is a read-only property.

5.11.4.2.4 ActiveLowAlarms

Indicates the number of active low severity alarms. This is a read-only property.

5.11.4.2.5 ActiveLowNACKAlarms

Indicates the number of unacknowledged low severity alarms. This is a read-only property.

5.11.4.2.6 ActiveMedAlarms

Indicates the number of active medium severity alarms. This is a read-only property.

5.11.4.2.7 ActiveMedNACKAlarms

Indicates the number of unacknowledged medium severity alarms. This is a read-only property.

5.11.4.2.8 ActiveNACKAlarms

Indicates the total amount of unacknowledged alarms in the application (active or not). This is a read-only property.

5.11.4.2.9 BackupDiscardInterval

Indicates the amount of time units during which backup data is kept until it is discarded. This property works along with **BackupDiscardTimeUnit** property. Default value is 12 (twelve time units indicated by **BackupDiscardTimeUnit)**.

NOTE: The total amount of time units indicated by the combination of the **BackupDiscardInterval** and **BackupDiscardTimeUnit** properties must be longer than the time indicated by the **DiscardInterval** e **DiscardTimeUnit** properties.

5.11.4.2.10 BackupDiscardTimeUnit

This property indicates the time unit to keep backup data until it is discarded. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

This property works along with the **BackupDiscardInterval** property.

5.11.4.2.11 DataSource

A Defines the Database to be used to record alarm data. Default value is an empty **String**, that is, there is no Database to store data.

5.11.4.2.12 DiscardInterval

9 This property works along with the **DiscardTimeUnit** property. The **DiscardInterval** property indicates the total amount of time units during which alarm data is stored in the database table, until it is discarded. Default value is 1 (one time unit indicated by the **DiscardTimeUnit** property). If this property is configured with a value less than or equal to the **BackupDiscardInterval** property, E3 automatically sets the value of **BackupDiscardInterval** as doubles the value of **DiscardInterval**. This property can be changed at run time.

5.11.4.2.13 DiscardTimeUnit

This property works along with the **DiscardInterval** property. The **DiscardTimeUnit** property indicates the time unit to store data on a table, until it is discarded. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

5.11.4.2.14 EnableBackupTable

◩ Creates a backup table where discarded data is stored, for security reasons. If True, the table is created. Otherwise, it is not. Default value is True.

5.11.4.2.15 EnableDiscard

◩ Indicates alarm's data discard after a certain time. If False, data is stored indefinitely on the table. Otherwise, it is not. Default value is False.

5.11.4.2.16 InactiveHighNACKAlarms

9 Indicates the number of active and unacknowledged alarms with high severity. This is a read-only property.

5.11.4.2.17 InactiveLowNACKAlarms

9 Indicates the number of active and unacknowledged alarms with low severity. This is a read-only property.

5.11.4.2.18 InactiveMedNACKAlarms

9 Indicates the number of active and unacknowledged alarms with medium severity. This is a read-only property.

5.11.4.2.19 InactiveNACKAlarms

9 Determines the total amount of active and unacknowledged alarms. This is a read-only property.

5.11.4.2.20 Logging

◩ Creates an alarm information record on the Database, specified by the **DataSource** property. If False, record is not created. Otherwise, it is. Default value is False.

5.11.4.2.21 TableName

A Defines a name for the alarms table. Default value is "Alarms". It can be changed at run time, and it is applied immediately.

5.11.4.2.22 VerificationInterval

9 This property works along with the **VerificationUnit** property to control the time interval E3 checks if data is old, and then discard it. Default value is 1 (one time unit indicated by the **VerificationUnit** property).

5.11.4.2.23 VerificationUnit

This property works along with the **VerificationInterval** property. The **VerificationUnit** property indicates the time unit to check for data discard. The available options are:

- **0 - dtHour**: hours
- **1 - dtDay**: days
- **2 - dtMonth**: months (default)
- **3 - dtMinute**: minutes

CHAPTER 6

Frequently Asked Questions

How to make a windowed Screen display a title bar with Minimize, Maximize, and Close buttons?

To do so, users must use Splitter's **SetFrameOptions** method. The *Flags* parameter specifies window features. A value of 127 defines a window with visible **Minimize**, **Maximize**, and **Close** buttons.

How to open a modal Screen?

To open a modal Screen, use Viewer's **DoModal** method. For example:

```
Application.DoModal "Screen1", "Title1", 0, 0, 400, 200, 0, 1
```

This code opens a Screen named `Screen1`, with title "Title1", at position (0, 0), with 400 pixels width and 200 pixels height, passes 0 as a Screen parameter, and enables a window's title bar.

How can I copy values from an E3Browser's row to a Tag?

First, select the row (or record) in E3Browser. Then, use E3Browser's **GetColumnValue** method. The *Index* parameter is the column's index to copy (starting at 0).

How to prevent users from typing a String in a SetPoint?

Check if the typed value is a number on SetPoint's **Validate** event. For example:

```
Sub Text1_Validate(Cancel, NewValue)
  If NOT IsNumeric(newValue) Then
    MsgBox "The value must be a number."
    Cancel = True
  End If
End Sub
```

How to open a date picker to select a date and a time when clicking a SetPoint?

By using the **ShowDatePicker** method on SetPoint's **Click** event. For example:

```
Sub Text1_Click()
  Dim datevalue
  If Application.ShowDatePicker(datevalue) Then
    Value = datevalue
  End If
End Sub
```

How to acknowledge all alarms of an Area?

To acknowledge all alarms of an Area by script, users can use the **AckArea** method. This method has two parameters:

- *Area* is the name of the Alarm Area that the alarm should be acknowledged
- *User* is the name of the logged on user, which can be the **Application.User** item

To acknowledge all active alarms, users can also use the **AckAllAlarms** method.

How to execute an action when clicking a specific mouse button or key?

By using Screen's **KeyDown** or **KeyUp** events. These events are triggered when a key is pressed or released, and return two parameters. One is the ASCII code of the pressed key, and the other one is the SHIFT and CTRL key status when the key was pressed. The idea is to compare event's return parameters with the ASCII code of the expected character.

How to create a WhileRunning script?

By creating an event linked to a property that always has the same value. For example, the **Visible** property of a Screen object. While the object is visible (its **Visible** property set to True), the script is executed. However, it is recommended to avoid **WhileRunning** scripts, because they may affect application's performance. In most cases, they can be replaced by Links.

How to create an OnValueChanged script?

By creating an event linked to the Tag's **Value** property, which is executed when a property changes its value.

NOTE: Be careful to not use Viewer methods on Server, as for example the **MsgBox** method. If this is the case, the event must be created on the Screen or on the Viewer object itself, instead of creating it on the Tag.

How to create Tags and Screen objects at run time?

By using the **AddObject** method. For example, the following script creates I/O Tags on `Driver1` Driver.

```
Set obj = Application.GetObject("Driver1")
For i = 1 To 100
  Set tag = obj.AddObject("IOTag", false)
  tag.Name = "IOTag" & CStr(i)
  tag.Activate
Next
```

How to display a message on Screen when changing a Tag's value?

By creating an event on the Screen linked to Tag's **Value** property, which is executed when the property changes its value. In this event, use the **MsgBox** method to display that message.

How to perform a date-filtered Query before assembling a Report?

To perform this, users must configure the Query object (see the **Query** chapter), which accompanies the Report, and then create the necessary variables on the **Filter** column. In the event that calls the Report, use a script such as the following:

```
Set report = Application.LoadReport("[Report1]")
Set query = report.Query()
query.SetVariableValue "Variable1", Value1
query.SetVariableValue "Variable2", Value2
report.PrintPreview()
```

Where:

- ***[Report1]*** is the name of the Report to call
- ***Variable1***, ***Variable2*** are the variables created on **E3TimeStamp** field's filter
- ***Value1***, ***Value2*** are the dates to query

To check other filter types, see the **Query** chapter or the available documentation at **Elipse Knowledgebase** website.

How to debug Server and Viewer script errors?

If the event is executing on the Viewer, use the **MsgBox** method. If the event is executing on the Server, use the **Trace** method.

Headquarters
Rua 24 de Outubro, 353 - 10º andar
90510-002 Porto Alegre RS
Phone: +55 (51) 3346-4699
Fax: +55 (51) 3222-6226
E-mail: elipse@elipse.com.br

USA
2501 Blue Ridge Road, Suite 250
Raleigh - NC - 27607 USA
Phone: +1 (252) 995-6885
Fax: +1 (252) 995-5686
E-mail: info@elipse-software.com

Taiwan
9F., No.12, Beiping 2nd St., Sanmin Dist.
807 Kaohsiung City - Taiwan
Phone: +886 (7) 323-8468
Fax: +886 (7) 323-9656
E-mail: evan@elipse.com.br

Check our website for information about a representative in your city or country.

www.elipse.com.br

kb.elipse.com.br

elipse@elipse.com.br

Microsoft Partner
Gold Independent Software Vendor (ISV)

OPC FOUNDATION MEMBER